Australian Government
Department of Defence
Defence Science and
Technology Organisation

DSTO

Modelling Intention Recognition for Intelligent Agent Systems

Clint Heinze

DSTO-RR-0286

Australian Government
Department of Defence
Defence Science and
Technology Organisation

Modelling Intention Recognition for Intelligent Agent Systems

Clint Heinze

Air Operations Division
Systems Sciences Laboratory

DSTO–RR–0286

ABSTRACT

Mainstream visual psychology presents a 'sense then infer' account of vision that is analogous to the 'sense then infer' processing that characterises the agent intention recognition literature. From ecological psychology comes Gibson's theory of visual perception that highlights the importance of the environment in explaining the nature of vision and recognition and claims that higher order structures are directly accessible. This theory can be used as the stepping-off point for an account of intention recognition and the means by which it might be modelled. Furthermore, the capacity for virtual environments to be designed 'agent friendly' provides yet another dimension of design freedom. When accompanied by an explicit model of perception the intention recognition problem can be cast as a software design problem. The resulting design patterns provide useful options for modelling intention recognition in intelligent agent systems.

Published by

DSTO Systems Sciences Laboratory
PO Box 1500
Edinburgh, South Australia, Australia 5111

Telephone: *(08) 8259 5555*
Facsimile: *(08) 8259 6567*

AR No. 013-251
November, 2004

APPROVED FOR PUBLIC RELEASE

Modelling Intention Recognition for Intelligent Agent Systems

EXECUTIVE SUMMARY

This report is an *almost* verbatim copy of Clint Heinze's PhD thesis of the same title conducted at the Intelligent Agent Lab at the University of Melbourne. This thesis was supported by DSTO from August 1997 – August 2003 under the Air Operations Research Branch's Long Range research task (currently LRR 03/226) and was supervised by Dr. Simon Goss. Some trivial typographic and layout changes have been made to support DSTO formatting requirements. It is published here to provide easier access throughout DSTO.

When constructing military simulations that require sophisticated cognitive models there are plenty of challenges to occupy the developer. Human factors experts, particularly cognitivists, endeavour to gain an understanding of actual human psychology behind the behaviours to be modelled; computer scientists attempt to develop computational technologies with the attributes necessary to model those cognitive functions. Ultimately engineers must draw these models and technologies together in a manner that results in simulation systems that meet the expressed requirements. In the world of military simulation requirements vary substantively. High fidelity, whilst seemingly always desirable often conflicts with performance, maintainability, complexity, and other attributes that combine to make the software less useful. Balancing these requirements is aided by the availability of several architectures that allow the engineer the freedom to tradeoff various attributes of the system.

This thesis presents a series of design patterns that provides software architectures useful for implementing intention recognition. Each of these architectures has a basis (although sometimes this tenuous) in psychology and cognitive modelling, and each imposes requirements on the technology necessary for implementation. The relative merits of the patterns are presented as are fully worked examples of their application to flight simulation.

Author

Clint Heinze
Air Operations Division

Clint Heinze is a cognitive scientist interested in the research spaces where software engineering overlaps AI. Primarily this involves attempts to discover ways of engineering representations of intelligence that have the properties commonly desired of quality software - that is they should be robust, reliable, validated, etc. He has a degree in Aerospace Engineering from RMIT and has recently completed his PhD in the Department of Computer Science and Software Engineering at the University of Melbourne. Since 1989 the Defence Science and Technology Organisation have employed him as a defence analyst and software engineer. In this capacity he is responsible for a range of activities associated with developing, employing and deploying smarter computer generated forces for military simulation in support of operations research.

Contents

Figures

Chapter 1

Introduction

> *"His plans are calm and deeply hidden, so no one can figure them out. He changes his actions and revises his plans, so that people will not recognize them. He changes his abode and goes by a circuitous route so people cannot anticipate him. When people never understand what your intention is, then you win."*—Sun Tzu [174]

The quotation above comes from the Chinese philosopher and writer, Sun Tzu. Though written around 500BC, the *Art of War* is still highly regarded for the insights it offers into military strategy. Two and a half thousand years later intention recognition is still a vital aspect of military decision-making: when intention recognition is successful the element of surprise is removed and the enemy successfully anticipated[1]; when intention recognition fails the results are often catastrophic[2].

Although significant in military contexts, everyday life is full of examples of the importance of intention recognition. Predicting and anticipating the behaviour of others eases interactions and improves cooperation and coordination. The simplest explanation for the need for intention recognition lies in an inability or unwillingness to communicate. If someone is both willing and able to communicate fully and honestly about their current and future actions then there is little need for intention recognition. If communication is unavailable, unreliable, expensive, impractical, or, as is the case with combative or competitive environments, undesirable, then a mechanism for interpreting and predicting the behaviour of others is required. Intention recognition goes to the very heart of the ability to display intelligent behaviour in environments where communication is impaired or impractical.

The original motivation for this thesis stemmed from exactly the point where Sun Tzu's statement about the importance of understanding intent intersects with virtual environments and intelligent agents—military simulation. In virtual environments intention

[1] And clearly the reverse is also true. By manipulating and deceiving the opponent with false indications of intent military victory can be assisted. Operation Bagration was an attack by 166 divisions of the Red Army through Belorussia and into Minsk. In perhaps the greatest defeat of the Germans during the second world war the Soviets successfully employed strategic level deception and convinced the German High Command that attacks should be expected elsewhere. The result was an unprepared and under-equipped German force taken almost completely by surprise.

[2] In 1988 the American warship, USS Vincennes misidentified and misinterpreted the actions of an Iranian Airbus carrying 290 civilian passengers. The result was an order to shoot and a tragic loss of human life.

recognition is as important as it is in the real world. Proliferating heterogeneous computer systems, networks, and agents that speak different languages with different purposes and designs require substitutes for communication. Intention recognition is a viable, logical, and natural alternative[3].

The wider information technology (IT) community has a clearly expressed need for models of intention recognition. Games, digital assistants, wizards, e-business intermediaries and broking agents could all benefit from models of intention recognition.

> *"As connected devices proliferate, you'll want software agents to help coordinate them all. Software agent developers hope to have agents evolve from mere facilitators into actual decision makers. But don't expect to have truly smart* ***agents that anticipate your needs*** *and negotiate on your behalf—at least not in the next five years."*—J. Clyman The 5 Developments to watch in the next century. PC Magazine, August, 2000

Interactive systems might apply models of intention recognition to enable computers to better assist the human user. Used in this way intention recognition can bridge the communication gap that exists between the human and the computer and provide a means by which the computer obtains predictive insights into the expected future behaviour of the user, preempting their requests with timely provision of information and services.

Because this thesis explores issues related to intelligent agents and the development of systems that include them, before embarking upon a summary of the thesis a definition of 'agent' is presented:

> **an agent is an autonomous entity situated in an environment that it can perceive, and in which it can act.**

Whether or not this is a useful definition of agency is explored in later sections but it is an acceptable starting point for the thesis[4]. Providing the definition immediately begs questions about the meaning of words like 'autonomous', 'situated', and 'environment'. Leaving aside the detail, it constitutes a reasonable working definition with which to commence the elaboration of the content of the thesis and a discussion of the significance of these terms is delayed for later chapters.

Implicit in the definition above is the widely held view that abstraction is an important property of agency. By developing software based on more abstract concepts than those available in object oriented programming[5] or procedural programming[6] previously intractable software developments are made tractable. Abstract concepts that have proved useful in agency are those that match human behaviours: such as perception and action.

[3]Intention recognition is only a substitute for communication as it pertains to coordinated activity. There are many other subjects of communication that are not subsumed by intention recognition

[4]This definition is broad enough to include both those agent architectures that are decomposed by function and those that are decomposed by activity [13]. Though Brooks' proposals for non-representational intelligence results in robots that display intelligent behaviour it is not clear that modelling and designing high-level behaviours like intention recognition is possible in such a framework.

[5]Where classes, objects, methods, data, and messages are the useful concepts.

[6]Where functions, subroutines, parameters and modules are the useful concepts.

The approach adopted throughout this thesis is predicated on the assumption that the word 'perceive' in the above definition is significant. The thesis presents insights into the design of agent systems related to the concept of perception. Though a recurring theme in definitions of agency, few theories, architectures or languages provide useful descriptions of perception. Most agent researchers seem to agree that perception is a worthy concept but few seem to take the step of explicitly including it into their theories, methodologies or languages. The situated nature of agents suggests certain models of perception—those from ecological psychology and the situated cognition literature [57, 30].

A simple definition of perception is presented here, broad enough to encompass at least some aspects of intention recognition:

> **perception is the process by which an agent becomes aware of events and structures in the environment**

Incorporating an explicit model of perception into the design process allows for a more reasoned approach to the development of agent systems, particularly those aspects related to interactions between the agent and its environment. Simply put, the design issue for agent systems becomes a question of what events and structures should be in the environment, and how the agent becomes aware of them. Mainstream theories of human vision posit that perception is a two-stage process: first the environment is sensed; then the *sense-data* is abstracted or otherwise processed into inferred higher order structures. This 'sense-and-infer' process assembles more abstract concepts from the raw material of sensation by an internal inferencing process.

More radical theories of visual perception propose that abstract structures are *in the environment* and are *directly accessed* by perception. Pattern matching techniques might search for and discriminate the patterns in the environment that are the indicators of higher order structures. Furthermore, in virtual worlds the environment can be designed. If the structures are there, hidden just below the surface but accessible and waiting to be perceived, then they can be made explicit by something akin to a labelling process. This transforms the task of perception from complex pattern matching into the relatively trivial task of label reading, or *sensing* as it will be referred to for the remainder of this thesis.

The following definition:

> **intention recognition is the process by which an agent becomes aware of the intent of others**

draws a deliberately strong comparison with the previous definition of perception. Perception involves the production of abstract representations of the environment from the rich array of perceptual input. Intention recognition is the most abstract form of perception-based human behaviour. Not only does it suggest *model of other mind*, but it is predictive, requiring representation of future states. Intentions of other agents cause patterns of events and structures in the environment and if these are *perceived* then the result of that perception *is* intention recognition.

Dennett argues convincingly that intention ascription is the means by which folk-psychological understanding and prediction of complex behaviour is achieved [39]. A recent paper by Isla and Blumberg argues for the application of Dennett's *Intentional Stance* to the development of artificial intelligence but leaves unanswered questions addressed by this thesis.

> *"[...]one of the important aspects of the intentional stance (Dennett's take on theory of mind) is that it can occasionally be applied fruitfully to objects that we know do not have minds of their own—the point being that as long as the attribution of intentionality is useful in predicting the objects behaviour, then it is as good a model as any. Initially, we, expect it will be convenient to tag explicitly objects in the world for which a theory of mind should be assumed. However, it would be interesting in the long-run to see whether a character can learn which types of objects can be so treated."*—Isla and Blumberg [81]

Current intelligent agent implementations of intention recognition adopt a 'sense then infer' approach analogous to mainstream theories of human perception [15]. Consequently these implementations are focused on the addition of inferencing, deductive reasoning, hypothesis generation or other, similar functionality to agents. These implementations provide intention recognition at the expense of modelling difficulties in other areas of the agent system, often significantly increasing design complexity. Furthermore, these implementations typically provide useful, sophisticated and interesting approaches for dealing with the inference component of sense and infer but have less to say about the manner in which the environment is sensed.

Developers of agent systems often fail to consider the *situated* nature of agents and the corresponding design interplay between the environment and the agent. Neglect of the environment is sometimes due to system constraints that place the environment beyond the design scope, but sometimes to a tendency for agent developers to disregard or undervalue those areas of the system that are 'non-agent' and to design their agents in isolation, or in idealised or simplified environments. In virtual worlds (or even in real-worlds where there is some design flexibility) the agent-environment interaction can explore a design space that provides for varied approaches to modelling intention recognition. Facilitating this exploration is a modelling and methodological task that is informed by the lessons of software engineering and the scientific research underpinning the philosophical and psychological literature relating to perception and recognition.

1.1 Aim and Scope

The aim of this research is to provide the designer of agent systems with a practical approach to modelling intention recognition for a range of intelligent agent systems. This can be decomposed into two related aims: to provide an approach to modelling intention recognition for intelligent agent systems; and to demonstrate that when undertaking the design of intelligent agent systems this modelling approach offers software engineering advantages.

To achieve the stated aim the research leverages on the experiences of mainstream software engineering in dealing with modelling and design issues—specifically the development and use of software design patterns. By adopting a software engineering approach this thesis does not advocate a particular agent architecture in which intention recognition might be modelled, nor is it limited to one particular technology that is suitable for recognising intent, or a logical account of a particular theory of intention recognition (though examples of each of these is provided). Instead, an approach to modelling agent systems is presented that allows for the intention recognition design space to be explored in the light of system requirements. This exploration results in a set of six architectural design patterns that provide the basis for modelling intention recognition in substantially different ways. These design patterns provide the software engineer with a tool-box of options to meet the varying requirements of systems that will be encountered in practice. To demonstrate the utility of these design patterns an intelligent agent system (the Virtual Air Show) is described that draws out the differences between the patterns and allows a critical evaluation of their strengths and weaknesses.

By considering *agent systems* (as opposed to agents in isolation) this research specifically includes the environment into which the agent will be placed as an important consideration in the design process. Design options result from allowing flexibility in the location of information and processing within the agent system and these are reflected in the developed patterns. In some cases this approach moves functionality that might normally be regarded as perception from the agent into the environment, and some of the processing that might normally be regarded as cognitive reasoning into perception. Objections to this will abound. Chalmers et. al. [26] caution AI researchers against drawing strong distinctions between cognition and reasoning. Brooks and others [13] offer similar criticisms about ignoring the interfaces between AI software and the environment. A critical assessment of the chosen modelling approach examines some philosophical and psychological implications in addition to the pragmatic realities of an engineering evaluation.

In seeking inspiration for these patterns the agent literature, the psychology literature and the software engineering literature are consulted. This thesis draws on the psychology literature for design inspiration but software engineering utility is ultimately a stronger driver than the psychological credibility of an adopted model.

The complexities resulting from the social interactions of multi-agent systems are specifically excluded from the scope of this thesis. In Chapter 8 the issue is discussed briefly with respect to possible extensions of this research.

It is a fundamental assumption of this thesis that the agents under consideration manipulate abstract, knowledge level data [125]. This assumption is one commonly associated with agents in general, intelligent agents in particular, and is unavoidable when considering agents that must exhibit high-level human behaviours like intention recognition. A related assumption is that the agent is situated in a complex environment where data from many sources (agents and non-agents) must be processed.

Casting intention recognition in intelligent agent systems as a modelling and design activity provides a solid software engineering basis for use in industry, practical insights for the application of technology, and, as Herb Simon writes:

> *"What is there to study besides the boundary sciences—those that govern the means*

and the task environment? The artificial world is centered precisely on this interface between the inner and outer environments; it is concerned with attaining goals by adapting the former to the latter. The proper study of those who are concerned with the artificial is the way in which that adaptation of means to environments is bought about—and central to that is the process of design itself."—Herb Simon [161]

1.2 Thesis Contribution

There are two primary contributions of this thesis that correspond to the two constituent parts of the stated aim:

- a set of six design patterns to support the designer of intelligent agent systems that require an intention recognition capability. These patterns are presented in the style of the mainstream software engineering literature [18] but are appropriately inspired by a consideration of the psychology literature; and
- a description of an implemented system that illustrates the application and utility of these design patterns and provides the basis for a critical appraisal.

There are two secondary contributions that were a by-product of the particular methodology adopted. In seeking inspiration for the design patterns the agents literature, software engineering literature and the psychology literature were examined. This gave rise to:

- an account of the importance of perception in modelling agent systems. In particular, the importance of an explicit model of perception as the means by which structures or concepts in the agent's environment are converted into representations that are appropriate for the agent; and
- a description of the manner in which ontologies assist intelligent agent system design and the relationship between perception and ontologies. Influenced by the developed model of perception an account of ontologies is presented that describes a means of integrating agent ontology design into mainstream software engineering. The ontology is seen as a product of the design or, if it is preexisting then a constraint over the design.

Finally there are implications for the general application of the approach described in this thesis into other related areas of agent design and of agent behavioural modelling. The wider applicability of this research and the future directions that it might take are discussed in Chapter 8.

1.3 Historical Influences

This thesis was motivated by specific requirements from the domain of military simulation set in the context of a more general, and ultimately more difficult, goal of advancing

the field of agent system design. This section, and the next, introduces some of these motivating factors which set the context for the thesis.

This thesis was originally conceived in the Defence Science and Technology Organisation (DSTO)[7] to meet a need to provide intelligent agents used as human surrogates in military simulations [76, 78, 74] with a capacity for intention recognition. This thesis is part of a larger ongoing research program and the need for intention recognition is just one of many in a long 'shopping-list' of agent functionality that the military simulation community requires [140].

Current generation simulators do not yet exhibit an intention recognition capacity but a number of prototypes and technology demonstrators have broken new ground and made significant advances. The addition of intention recognition to agent based military simulations promises to greatly enhance their utility in exploring issues associated with command and control and tactical decision-making and to generally improve the fidelity of computer generated forces in human-in-the-loop training simulators [115].

Intention recognition was identified by Rao and Murray[8] as one of the significant impediments to higher fidelity modelling of human decision-making in military simulation.

> *"The tasks performed by a combat pilot can be broadly divided into two areas—situation awareness and tactics selection. Situation awareness is defined as the knowledge, understanding, cognition, and anticipation of events, factors, variables affecting the safe, expedient and effective conduct of an air mission. ... In obtaining situation awareness, a pilot must infer the beliefs, desires, and intentions of other pilots from the behaviour of their aircraft."*—Rao and Murray [151]

Their initial insights led to further work in developing a theoretical basis for modelling intention recognition [147]. This was taken by Busetta and Tidhar and operationalised, in the form of a technology demonstrator [19][9]. Initial experiences with the technology, though positive, revealed a number of unresolved challenges. It was clear that traditional approaches to intention recognition, though often elegant in their theoretical formulation place limitations, constraints, or burdens on other areas of the agent design. In an environment where performance is critical, continuous pressure for the computationally light solutions dictates that approaches like those of Rao and Murray are not always preferred.

Beyond the task of providing approaches for modelling intention recognition in intelligent agent systems there are secondary motivations from the domain of military simulation that have pushed, pulled, and otherwise directed this thesis. This brings together some of the strands of a decade of research related to agent-based military simulation [74, 169, 114, 170, 78, 76], agent oriented software engineering [70, 73, 136, 77, 75], and cognitive modelling [71, 72, 126].

DSTO uses agent based simulation to answer questions that are impractical to answer otherwise. Military systems are associated with long development schedules: often many

[7]The DSTO provides science and engineering advice to the Australian Defence Force.

[8]The work of Rao and Murray and the subsequent research by Busetta and Tidhar was undertaken with the support of DSTO.

[9]Further details are included in Section 2.3

years will separate the decision to purchase a piece of equipment with the date it will enter service. Simulation provides the only realistic option for evaluating systems during that interim period. Even if the actual hardware has been acquired and is available for experimentation it is also true that military systems are notorious for their high costs and the potential hazards associated with their operation. Simulation provides a cost-effective means of experimenting with the hardware in a safe environment.

There is a natural demarcation in military simulation between the models of physical systems (military hardware, aircraft, ships, radars) and models of the personnel who operate them (pilots, soldiers, seamen, commanders). The former are well understood, relatively simple to model and are usually simulated with standard software approaches. The human component is problematic. Decision-making is difficult to model, is influenced by a large number of variables and colored by emotions, experience, memory and other human factors. Agent-based models can provide solutions to at least some of these models. DSTO is actively investigating areas such as command and control, cognitive systems engineering, and the theories of naturalistic decision making to improve the computational modelling of military personnel. The broader research program will bring this diverse research (including this thesis) together with the ultimate goal being a unified model of human cognition in the style of Newell [124] but with equal weight given to issues of software engineering, social and organisational modelling, and knowledge engineering.

Advice provided by DSTO may impact on multi-billion dollar defence acquisitions and on the safe conduct of operations. Robust, reliable, high performance, well validated and maintainable software is essential and is sometimes at odds with the need for flexible, high fidelity models of human decision-making. With a substantial research and development investment in intelligent agent technologies and a significant percentage of total code developed being agent oriented DSTO has a vested interest in the future of relevant software engineering methodologies.

This thesis arose from a domain where the need to provide models of intention recognition for intelligent agents in simulation environments was influenced by: the high priority of software engineering; the performance dictates of real-time and faster than real-time simulation; the need for a range of solutions; the importance of validation and verification; and the development of a cognitive model.

1.4 Agent System Design Influences

This thesis does not aim to explore agent system design in general but the results of this thesis may inform a broader research community than that suggested by the specific title. A motivating driver for this research was the continued investigation of methods, techniques, and tools that can support the broad spectrum of agent software design activities. Within the broad spectrum of software engineering activities classifiable as 'design' there are four that are related to this thesis.

Agent Interface Design Most software bugs occur at module interfaces [178]. This is particularly true of agent systems where interfaces translate data from the standard software that is typical of agent environments into the novel languages, higher levels

of abstraction, and ontologies that are typical of agents [188]. Insights into methodologies for designing the interfaces between agents and environments are available by explicitly modelling perception [77]. The literature suggests that interfacing agents with environments is problematic. The interfaces are constrained by many factors: the nature of the environment, ontologies, requirements on the agent behaviour and other system requirements. These conflicting requirements demand an holistic view. An agent-centric view simply exacerbates the problem:

> *"When a researcher working on a particular module gets to choose both the inputs and the outputs that specify the module requirements I believe there is little chance the work they do will fit into a complete intelligent system"*—Rodney Brooks [13]

Designing Agent Friendly Environments Agents, though autonomous and modular, must be designed within the constraints of the environments in which they are situated. In domains where the environment is within the design scope there is an interplay between the design of the agent and the design of the environment. Designing environments so that they are *agent friendly* is an important agent engineering issue that is informed by a consideration of agent perception [135]. Awareness of the importance of the environment in shaping the design of agents has been recognised by some researchers.

> *"Without an environment, an agent is effectively useless. Cut off from the rest of its world, the agent can neither sense nor act. An environment provides the conditions under which an entity (agent or object) can exist. It defines the properties of the world in which an agent will function. Designing effective agents requires careful consideration of both the physical and communicational aspects of their environment."*—James Odell et. al. [133]

Environments that agents are placed into are often legacy systems lacking the necessary features. Strategies for overcoming the limitations of these environments are required for allowing the proliferation of agents into real-world settings.

Utilising Ontologies for Agent Design Agents are typically 'knowledge-level' entities. Certainly those that will exhibit abstract reasoning like intention recognition qualify. The growth of interest in ontologies promises to provide guidance for structuring and sharing agent knowledge. Research into agent ontologies is split into two broad camps. The global, sharable ontology community and the lightweight ontologies community. In both cases there are requirements to design use and reuse ontologies [63, 193, 65, 180]. Furthermore it is desirable to incorporate the design and development of ontologies into the software engineering process either as constraints (in the case of a predefined reused ontology) or as a set of requirements (in the case of an ontology to be developed).

> *"[...] the success of these efforts depends on the development of an engineering discipline for ontology design, akin to software engineering for conventional software."*—Gruber [63]

Developing Design Patterns for Agents Patterns provide powerful examples of encapsulated experience that allows software engineers to reuse the successfully applied designs from previous developments. A set of appropriate modelling choices and design patterns that characterise the spectrum of possible intention recognition approaches will guide the development of future intelligent agent systems.

> *"Developers of AI software are normally faced with design challenges involving robustness, efficiency, and extensibility. Most of these challenges at a higher level are independent of the application-specific requirements. Although design patterns have been successfully adopted to tackle these issues, they are rarely documented. Consequently this knowledge remains hidden in the minds of developers or buried within complex system source code."*—Kostiadis et. al. [100]

These aspects of design have three things in common: they been identified in the literature as deficiencies in the state of the practice of intelligent agent development; they are integral to an engineering approach to modelling intention recognition; and they are issues that are being addressed currently by those researchers concerned with agent oriented software engineering. It seems clear that agent oriented software engineering techniques will increasingly become important tools for the software engineer.

> *"It is already a good bet that the software engineering of tomorrow will be 'agent oriented' just as that of today is beginning to be object oriented"*—Jacques Ferber [48]

Many agent oriented analysis and design methodologies have been suggested [17, 185, 182, 29, 97] and the breadth of research in this area has resulted in the emergence of many agent-related concepts that are useful for analysis and design. It is desirable to distill the important concepts to gain insights into more general agent development methodologies [89].

This thesis will not attempt to develop any general methodological approach but will, through example, detail some techniques and models that might be components of a more general approach to agent systems development. In adopting an agent oriented design approach to modelling intention recognition this thesis hopes to meet the challenge set by Nick Jennings:

> *"Agent-oriented techniques represent an exciting new means of analysing, designing and building complex software systems. They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled. Yet, to date, there have been few serious attempts to cast agent systems as a software engineering paradigm."*—Nick Jennings [84]

If agent oriented software engineering is to become a powerful, widely adopted, mainstream methodology then a process of unification, standardisation, and industrialisation must occur[10]. The disparate nature of agent research and technology means that unifying

[10]These processes are those that have been used to characterise the development of the UML and object oriented software engineering [130]

agent oriented software engineering might be problematic. The maturity of object oriented analysis and design and related tools, languages, and methodologies will be an important influence over a future unified view of agent oriented software engineering [89]. With this in mind the unified modelling language (UML) is adopted and extended here as a notation for describing and documenting the presented design [73, 136].

1.5 Structure of Thesis

This thesis is divided into four parts. Part introduces and summarises the thesis by articulating the aim and scope and the contributions of the thesis then providing an historical background to situate the thesis in a broader context. Chapter 2 then discusses in more detail the related literature and relevant theoretical and practical approaches that provide the necessary background.

Part 2.8 directly addresses the first part of the aim of the thesis "to provide an approach to modelling intention recognition for intelligent agent systems". Together the four chapters of Part 2.8 provide the methodological part of the thesis resulting in a set of design patterns for modelling intention recognition in intelligent agent systems. Chapter 3 examines three ways in which perception influences models of intention recognition. In order to set a modelling framework in which to explain existing approaches and to provide the basis for further development an explicit model of perception is presented. Secondly Gibson's theory of direct visual perception is extended to explain intention recognition in humans. Gibson's original theory and its extension into intention recognition provides important insights for modelling intention recognition in intelligent agent systems. Finally, the importance of the environment in conditioning the design of agent perception is examined and a framework for modelling intention recognition in intelligent agent systems is developed. A byproduct of the adoption of an explicit model of perception is the elaboration at design time of the knowledge level concepts that are important to the agent. This establishes a strong link with existing research into ontologies. Chapter 4 completes the background theory necessary for developing the models of intention recognition by investigating the roles that ontologies can play in designing agent systems. Chapter 5 applies the lessons from Chapters 3 and 4 to develop a set of architectural designs that provide alternatives for modelling intention recognition. These designs are elaborated and presented in the style of the mainstream software design patterns literature in Chapter 6.

Part 6.3 addresses the second part of the aim of the thesis "*to demonstrate that when undertaking the design of intelligent agent systems this modelling approach offers software engineering advantages*". The systems described apply the design patterns that were the principal result of Chapter 6 to the design of six variants of an intelligent agent system. The developed system, a flight simulator with an intelligent agent acting as a flight-training instructor is sophisticated enough to provide an indication of the strengths and weaknesses of each of the design patterns. A single example (the most technologically challenging) is described in detail and is taken to implementation in Section 7.3.

Part 7.3.4.3 summarises and discusses the implications of this research and provides pointers to the possible extension and reapplication of this work.

The four identified contributions of this thesis correspond to the primary chapters:

Chapter 3 *An account of the importance of perception in modelling agent systems. In particular, the importance of an explicit model of perception as the means by which structures or concepts in the agent's environment are converted into representations that are appropriate for the agent.*

Chapter 4 *A description of the manner in which ontologies assist intelligent agent system design and the relationship between perception and ontologies. Influenced by the developed model of perception an account of ontologies is presented that describes a means of integrating agent ontology design into mainstream software engineering. The ontology is seen as a product of the design or, if it is preexisting then a constraint over the design.*

Chapter 5 and 6 *A set of six design patterns to support the designer of intelligent agent systems that require an intention recognition capability. These patterns are presented in the style of the mainstream software engineering literature [18] but are appropriately inspired by a consideration of the psychology literature.*

Chapter 7 *A description of an implemented system that illustrates the application and utility of these design patterns and provides the basis for a critical appraisal.*

Chapter 2

Background

"A large part of appearing intelligent is not only the ability to be predictable (in the sense of Dennett's intentional stance) but also the ability to form predictions, and to act in anticipation of those events predicted."—Isla and Blumberg [81]

"We'd been caring for the AI, weaning it. Then Donald warned us that the system started to display more than consciousness. It started to display intention."—X-Files The Kill Switch Episode

This Chapter provides details of the literature, technology, and associated developments that inform, motivate and otherwise guide this research. Often the importance lies in emerging trends in particular fields or in the relationships between subject areas. It provides the background knowledge that sheds light on why certain approaches were taken and justification for the models that are presented. Links back to this chapter are supplied throughout Parts 2.8 and 6.3 so the reader might choose to skip this Chapter, referring back to it only when necessary. A summary of this Chapter (Section 2.8) provides a set of statements interpretable as requirements, constraints and assumptions against which the results of thesis is evaluated in Part 7.3.4.3.

Section 2.1 presents a brief history of agents and develops a consensus based definition of agency that sets the scene for the remainder of the thesis. As definitions change over time there are emerging trends that point toward relevant areas of research.

Section 2.2 provides an account of the relevant literature that deals with intention and its impact on the development of agent systems. In keeping with Section 2.1 intentions are presented as an ascribed property of a system or as part of the description of a system.

Section 2.3 builds on Section 2.2 by adding a survey of the intention recognition literature. Specific examples from the literature are critically assessed and a stereotypical case study presented.

Section 2.4 One of the emerging trends in agent research is the situated nature of agents and the corresponding importance of environments. The situated cognition literature, particularly that which relates ecological visual perception is important to the intention recognition models developed later in the thesis.

Section 2.5 The last decade has seen intelligent agent research subsume much of what was previously artificial intelligence. Areas of mainstream AI offer insights into important considerations for intention recognition. Specifically the focus on hybrid systems, certain types of pattern matching and the movement toward a 'Situated AI'.

Section 2.6 The emphasis on the design-time interplay between the agent and the environment suggests that software design and the lessons of software engineering will be necessary in anchoring the practical, methodological side of this thesis. Design patterns, important in the object oriented community have been identified as important in the development of intelligent systems.

Section 2.7 A discussion of ontologies can inform the design of agent modelling when, as is the case with intention recognition there are highly abstract (knowledge level) concepts involved, interactions between agents and system design issues to consider.

2.1 Agents and Agency

This section revisits the fundamental definition of agency, not to reopen what might be a rather futile debate, but to show that there are several important concepts in agency, perception being a significant one, that are common across all agent systems, are an integral part of agency, and will likely form part of a unified view of agency should one emerge. In the early 90's the OO community went through several years of unifying fragmented views before a clear and consistent approach to building OO software emerged. The agent academic research community has, in many respects, addressed the various definitional debates about agency and moved on. The multitude of views, definitions and opinions about agents is a property of the rich diversity of current research. This diversity, whilst healthy in a research community, can lead to difficulties in standards, pedagogy, and engineering practice. Without agreed common definitions about base concepts it is difficult to transition the lessons learned by the agent community into the classroom and industrial practice. So, though it is necessary to retain diversity to avoid stifling threads of research it is as important to move toward agreed definitions that can build a community of practice in industry. The definition of agency provided in Chapter 1 is not meant to stake a claim as *the* definition of agency. Rather it is indicative of the high level definitions that must be agreed upon if a unified view of agency is to be achieved.

It has been noted by Wooldridge and Jennings [188] that there are dangers for the agent developer in ignoring emerging standards. As experience with agent systems development grows, the successful reapplication of techniques, architectures, tools, and knowledge will require a unified view of agents. Efforts to standardise aspects of agency are proceeding in a number of quarters with the Agent UML [132], standard ontologies [63], and the work of FIPA [142] and the development of the KQML [49] being obvious examples. Those seeking to apply the results of agent research in an industrial setting must first ascertain the relationship between their model of agency and these evolving structures.

Specific types of agents require mobility, social ability, rationality, intelligence, and other properties but the generic definition above will cover the broad features. Some

might argue that the concept of proactivity or goal directed behaviour is important as it provides a clear distinction from objects that are generally considered reactive. Much of the research into agent theories is associated with defining, adding, and elaborating upon these *extra* features of agency. Elsewhere Juan et. al. [89] have addressed the problem of how to provide core definitions of agency with flexible variations for specific domains, agent theories, and languages much in the way that the UML provides profiles for particular modelling activities [7, 4].

It is still worth considering some of the different views of agent proposed over the last four decades—their common ground offers guiding insights for developers of agent technologies and methodologies. So from the foundations of agency in the 1950's comes the idea of agents as software servants answering human requests.

> *"The idea of an agent originated with John McCarthy in the mid-1950's, and the term was coined by Oliver G. Selfridge a few years later, when they were both at the Massachusetts Institute of Technology. They had in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a 'soft robot' living and doing business within the computer's world."*—Kay [93]

Notions of goal-directedness, autonomy, human surrogacy and knowledge-level interactions are clearly important even in this early definition and have recurred ever since. In this vein Wooldridge states that:

> *"An agent is an entity with four properties: autonomy, proactiveness, reactivity, and social ability."*—Mike Wooldridge [184]

Other views of agency, no doubt influenced more by embedded systems, robotics, cognitive modelling, and what Jordan and Russell call the "situated movement in AI" [88], strengthen the idea of agents as situated in an environment. Adding an environment to the mix introduces the idea that the agent must perceive to support its autonomous action.

> *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."*—Fanklin and Graesser [51]

> *"Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed."*—Patti Maes [109]

> *"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions."*—Barbara Hayes-Roth [69]

A more recent description of agency by Wooldridge agrees with these:

> *"An agent is a computer system that is situated in some environment, and is capable of autonomous action in that environment in order to meet its design objectives."*—Mike Wooldridge [186]

This shift observed over time is toward definitions that see agents as situated. mirrors similar related trends in AI research. This reflects the close links between AI and Intelligent Agent research. For many, the study of situated AI *is* the design of Intelligent Agents.

An important characteristic of agency that emerges in many definitions is the idea that agents are concerned with more abstract processing than other software[11]. They manipulate higher-level data and communicate in higher-level languages than other software. Nwana states this succinctly:

> *"Lastly. and perhaps most importantly. agent-based applications operate typically at the knowledge level (Newell, 1982), not at the symbol level as is the case in distributed computing applications. In any case, modules in distributed computing applications are not autonomous in the same sense as described earlier for agent applications. The majority of software applications may be ruled out from the set of agent-based applications on the same grounds that expert systems or distributed computing applications are."*—Hyacinth Nwana [129]

The idea that agents operate at a higher level of abstraction is important for the software engineering community where abstraction is a valuable tool in the management of increasingly complex software and is one of the more widely published advantages resulting from the adoption of agent technologies [74, 84].

Bradshaw makes yet another important distinction:

> *"Out of this confusion, two distinct but related approaches to the definition of agent have been attempted: one based on the notion of agenthood as an ascription made by some person. the other based on the description of the attributes an agent is designed to possess."*—Bradshaw [9]

The difference between ascribed and described agency will become an important aspect of dealing with intention recognition later in the thesis. Intentional states, important for modelling agent systems, are similarly capable of the same ascription/description demarcation.

Taking a software engineering view allows a partial reconciliation of Bradshaw's distinctly different views of agency (See Figure 2.1). From this perspective agents are ascriptions *and* descriptions depending upon the phase of the engineering process.

Further software engineering background is covered in Section 2.6 but the definitions of agency preferred by those agent researchers with a strong software engineering background and concerned more with the industrial application of agency are important in providing insights into the likely future of agents if they are to become a standard part of the software engineer's inventory of solutions. Jennings, for example, states the results of adopting an agent-oriented approach as:

[11] Even in the earliest definition of agency was the idea that agents should "ask for and receive advice, offered in human terms".

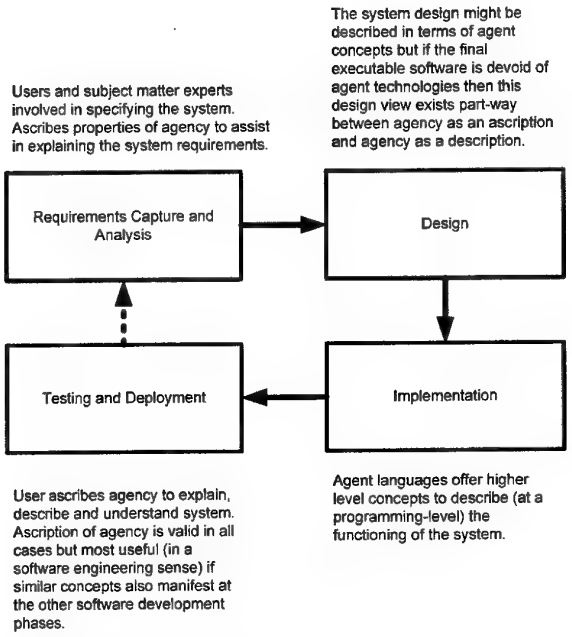

Figure 2.1: A Software engineering view of Bradshaw's distinction between ascription and description. If the agent concepts ascribed by the subject matter expert as they envisage or use the software match the agent concepts present in the design and implementation then the system is strongly agent oriented. This matching of structures in the head of the user (ascription) to those internal to the system (description) is a primary advantage of agent technologies and can facilitate through life validation and verification.

> *"the key abstraction models that define agent-oriented mindsets are agents, interactions, and organisations."*—Nick Jennings [83]

A similar emphasis on the social and interactional aspects of agency is provided by O'Dell and the other developers of the AUML [132] with the focus being on relationships between agents and not the development of approaches for the internal design of agents [96, 182, 185]. Agent software engineering development is being pushed by manufacturing process control, e-commerce and e-business, and web applications. The most pressing problems of this domain are the rapid proliferation of computer networks, typified by the spread of the web, that has generated legacy and integration problems for engineers tasked with connecting the disparate business systems of that were previously isolated. In addressing these issues it is the connections between agents, what Kinny et. al. calls the external model [96], that are more important than the detail of agent itself (the internal model). Though most agent oriented software engineering approaches concentrate on the agent aspects and provide little support for embedding agents into environments it is inevitable that as the field matures the environments in which agents are situated will receive increasing attention.

Returning to the definition of agency that was proposed in Chapter 1:

> **an agent is an autonomous entity situated in an environment that it can perceive, and in which it can act.**

Examining this definition in more detail:

autonomous An agent has the capacity to act by itself. The agent selects its own actions and there is an implication that the agent is proactive as well as reactive. There is a tradeoff between communication, autonomy and the social aspects of agency. It is expected that an agent can often decide for itself the appropriate course of action or discover the relevant information. In this respect agents are expected to be more independent than other software.

situated An agent is associated with an environment. It is linked with, via design, a particular environment and the association is two way. Agents influence the environment and vice versa.

environment For the purposes of this definition the environment includes everything external to the agent. This can, of course, include other agents.

perceive the agent has the capacity to perceive the environment. In simple software terms perception models the data entering the agent from all other components of the system. The important point here is that perception captures a sense of the proactive searching for data often associated with agents as well as the passive receipt of data from the environment. Describing the agent as having perception also carries something of the human metaphor that is a big part of the popularity and utility of agents. By this definition perception subsumes all of the possible inputs to the agent, including communication[12].

[12]It will be seen in later sections that treating communication as just another form of perception has some advantages during design.

act the agent can take action to change the environment. Again in a simple software engineering sense, action subsumes all of the output from the agent. Like perception this subsumes communication from the agent. Again the human metaphor plays an important role here in lending intuitive understanding about what it means to act.

Most proposed definitions of agency agree (or at least do not violently disagree) with this definition. Indeed this definition is deliberately broad and inclusive but there are some important points to note. The definition has aimed for wide coverage in the hope that most of the agent theories, architectures, and languages in existence can fit within this definition. The goal is a general definition that has broad coverage with specialisation available for particular theories, architectures, languages and domains. By being necessarily general it will need refinement and specialisations to handle particular classes of agent. The definition makes no claims about the social nature of agents and multi-agent systems. Researchers such as Tidhar point to the fundamental differences between agent systems and multi-agent systems [172]. This thesis does not directly deal with multi-agent systems. A discussion of the limitations of this thesis with respect to multi-agent systems is included in Section 8.3. Multi-agent systems are those where the agents have a social existence. This generally requires representing concepts such as identity, communication, and teams. Multi-agent systems are distinguishes here from populated single-agent systems where there are many agents but they have no concept of identity, they do not communicate, and they are aware of other agents only as observable entities in the environment that offer no particular opportunities for social interaction.

This definition specifically accounts for the inputs to (perceptions) and outputs from (actions) an agent. It is widely reported [178] and widely accepted that the greatest number of software development bugs occur at the module interfaces. By explicitly defining the agent interfaces as a part of the basic concept of agency a step toward addressing some of these issues is taken. The perceptions and actions also start to define the *signature* of the agent and will guide the developer in reusing the agents across multiple projects.

It is of course necessary to make some comment about the internal structure of the agent and it is here that other agent oriented software engineering methodologies and agent definitions make some strong claims to the nature of agency. Social aspects, mobility, learning, and beliefs, goals, roles, memory, and many other concepts have been used to define agency and recommended as the basic concepts of agents that lead into analysis and design methodologies. Because of the fragmented nature of agent research and development this thesis will try where possible to remain agnostic with respect to the existing technologies and to develop methods applicable for many.

2.2 Intention

Dennett [39], Searle [159], Anscombe [2] and Wittgenstein [183, 2] are just a few of the throng of philosophers that have published several works dealing with role intention plays in theories of consciousness, planned action, rationality and intelligence. Some philosophers are more concerned with the role that intention plays in directing rational decision making and guiding future action. For example, Bratman's work on rational agency in

the context of practical reasoning takes intention as the core attitude that directs future planning.

> *"Much of our understanding of ourselves and others is rooted in a commonsense psychological framework, one that sees intention as central. Within this framework we use intention to characterise both people's actions and their minds. [...] these characterizations provide a basis for our everyday attempts to predict what others will do, explain what they have done and coordinate our projects with theirs."*—Bratman [10]

Many agent researchers have recognised the importance of intentions in developing useful agent theories, architectures, and languages. Rao and Georgeff [149] developed a model of agency (BDI) that led to the commercialization of several high-level agent languages. Cohen and Levesque [32], Grosz and Kraus [61], and Konolige and Pollack [99] have each added valuable contributions to the growing literature dealing with the theoretical basis for incorporating theories of intention into agent research. The software engineering community is also starting to embrace the idea of intentions, though mainly with respect to user modelling for interface design [68] or forensics [87].

There are few proposals to use the concept of intention in an agent oriented software engineering context. In a thesis dealing with intention recognition it is to be expected that intention will play an primary role in the specification of an agent system. Intention recognition, whilst a useful and valuable functional capability for some agents, and the main thread of this thesis, is by no means universally applicable. Only a percentage of the agent systems under development might actually require an intention recognition capability. For those systems that require an intention recognition capability intentions are an important, even primary, component during specification, design and implementation. Stronger views regarding the intrinsic importance of intention to all forms of agency have been made. It has been previously claimed that:

> *"intention is a property of agency suitable for the analysis and design of all sufficiently complex agent systems regardless of their particular internal structure."*—Heinze [70]

By elevating intention to a fundamental concept of agency it assumes an important role in the development of an agent system. Exactly what role intention plays will depend upon how far through the software life-cycle the representation of intention can be carried. Just as agency itself can be an ascription or a description (see Section 2.1 and [9]) so too can intention be an *ascription*: a property of an agent useful to the specifier and user of the system; or a design or implementation level *description* of an agent useful to the designer and programmer. This thesis supports the view that for any sufficiently complex agent system regardless of the architecture or language chosen 'intention' is a fundamentally important concept. Intentionality can be an ascribed or described property of a system in precisely the same way that agency itself can be an ascription or a description at different stages of the software life-cycle. It has been observed by the developers of agent systems that embody folk-psychologic constructs that the intuitiveness and inherent familiarity that they offer simplifies aspects of software development. Whether or not the implemented system explicitly describes these constructs it seems probable that ascribing them might have utility during the earlier phases of software development.

2.2.1 Intention as Ascription

Ascription theory [160] has formed the basis for a number of important works in philosophy and psychology. Dennett, for example, argues convincingly that the ascription of intent is a perfectly reasonable, and indeed highly successful, way of predicting and describing the behaviour of systems that are complex enough to avoid explanation from other stances.

Dennett offers three stances that can be adopted to explain and predict the behaviour of systems. The physical stance operates entirely upon knowledge of physical composition and of the laws of physics. Those who advocate Laplace's deterministic universe might suggest that everything is explainable (at least theoretically) from this stance. Certainly many things are capable of being predicted by the laws of physics but:

> *"Sometimes, in any event, it is more effective to switch from the physical stance to what I call the design stance, where one ignores the actual (possibly messy) details of the physical constitution of an object, and, on the assumption that it has a certain design, predicts that it will behave as it is designed to behave under various circumstances. For instance, most users of computers have not the foggiest idea what physical principles are responsible for the computers highly reliable, and hence predictable, behaviour. But they have a good idea of what the computer is designed to do."*—Daniel Dennett [39]

There are systems that avoid explanation from the even the design stance however and for these Dennett advocates the intentional stance.

> *"Sometimes even the design stance is practically inaccessible, and then there is yet another stance or strategy one can adopt: the intentional stance. Here is how it works: first you decide to treat the object whose behaviour is to be predicted as a rational agent; then you figure out what beliefs that agent ought to have, given its place in the world and its purpose. Then you figure out what desires it ought to have, on the same considerations, and finally you predict that this rational agent will act to further its goals in light of its beliefs. A little practical reasoning from the chosen set of beliefs and desires will in many—but not all—instances yield a decision about what the agent ought to do; that is what you predict the agent will do."*—Daniel Dennett [39]

McCarthy summarises the position well.

> *"To ascribe certain beliefs, free will, intentions, consciousness, abilities or wants to a machine or computer program is* legitimate *when such an ascription expresses the same information about the machine that it expresses about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behaviour, or how to repair or improve it. It is perhaps never logically required, even for humans, but expressing reasonably briefly what is actually known about the state of the machine in a particular situation may require mental qualities, or qualities isomorphic to them. Theories or belief, knowledge and wanting can be constructed for*

> *machines in a simpler setting than for humans and later applied to humans. Ascription of mental qualities is* most straightforward *for machines of known structure such as thermostats and computer operating systems, but it is most useful when applied to entities whose structure is very incompletely known."*—McCarthy [113]

Adopting the intentional stance occasionally requires the ascription of intention to objects, agents, or entities without intent. It matters not, the intentional stance is still highly successful as a predictive tool: useful regardless of the internal structure. Russell and Norvig see this as a disadvantage:

> *"The "intentional stance" has the advantage of being based solely on the entity's behaviour and not on any supposed internal structures that might constitute "beliefs". This is also a disadvantage, because any given behaviour can be implemented in many different ways. The intentional stance cannot distinguish among the implementations."*— Russell and Norvig [156]

But in a software engineering context where *abstraction* and *implementation independence* are valued, the inability to distinguish from among implementations is actually a advantage. From a software engineering perspective ascribing intention will be most useful if there is some associated manifestation of that intention in the resulting software. Its utility in any case is that it provides intuitive insights into the workings of the system that accord with everyday folk psychological attributions that, according to Dennett are the human stock in trade for predicting the actions of complex systems[13].

2.2.2 Intention as Description

Some particular classes of agents, prototypically characterised by Rao and Georgeff's BDI agent model [149] or other cognitively inspired alternatives [99, 32], regard mental attitudes like intention as fundamentally important. Just as intentionality has an important role in the philosophical understanding of human behaviour there are a class of software agent systems that take intentions to be of primary importance in the description of agent behaviour. These agents typically include some manner of programming languages support for the representation, generation, and execution control of intentions. Examples include dMARS [42], UM-PRS [106], JAM [164], JACK [20], RIMA [11], and Attitude [104]. Programming support for representing intentions helps distinguish these languages them from agent systems that offer no such explicitness. A brief account of one of these languages follows. These agent systems provide the software engineer with the programming level tools necessary to describe agent behaviour in terms of intentions and related mental attitudes. For agents constructed from these languages an observer can still happily ascribe intentions to the agents but now these ascription can be grounded (if necessary) in concrete manifestations of intention within the executing agent.

[13]There is a corresponding view from developmental psychology where, it is argued by Olson et. al., there is a difference between ascribing intentional states to those that can also ascribe them to themselves (adults) and those that demonstrably cannot (children under the age of three) [191].

2.2.2.1 dMARS

dMARS is a multi agent architecture that implements a BDI model of agency based on the concepts of intentions, plans and practical reasoning developed by Bratman [10]. Further information about the underlying formalism of dMARS is available in [148] and [150].

Every dMARS agent comprises a set of beliefs, desires (goals), plans and intentions. The beliefs of an agent are stored symbolically in a relational database. Beliefs may refer to hard physical data sensed from the environment, more abstract concepts that are the product of some reasoning process, or representations of the internal state of the agent. The goals of an agent are descriptions of required behaviors or desired outcomes. The plans are graphical representations of procedural knowledge for specifying actions to take to accomplish these goals. The intentions are plan instances selected for processing to achieve a goal. They represent commitments by the agent to the achieving of a goal through the course of action specified by the plans. The graphical nature of plans allows them to be displayed during the simulation through the dMARS Control Interface. This feature allows the plans adopted by the agent to be displayed and the current state of the agent reasoning evaluated during the simulation. This gives observers visibility into the intentions actually adopted and executed by the agent. With careful design these plans can be read and understood by lay-people with little or no additional explanation. A detailed description of the dMARS system as it pertains to air-combat modelling can be found in [171] and further dMARS specific information may be obtained from [42].

dMARS has been used as the agent language for several large projects [37, 116, 151]. It has been chosen as the implementation platform for some of the agent components of the example systems described in Chapter 7.

2.3 Intention Recognition

In Chapter 1 the following simple definition of intention recognition was provided.

> **intention recognition is the process by which an agent becomes aware of the intention of others**

Before embarking on a discussion of the relevant background it is worth considering why intention recognition is necessary at all. Clearly there is a link between communication and intention recognition. Humans use both verbal and non-verbal queues to help ascertain the intentions of others. Developers of agent systems also make use of both verbal and non-verbal actions to provide insights into the reasoning of their AI [141].

> *"an actors internal state is made visible in three ways: language (spoken dialogue, even aliens can still convey mood, e.g. wort wort wort), different animation postures based on intention (sneaking, running, hiding), and specific animation triggers (gestures, canned dive/roll movements)."*—Chris Butcher and Jaime Griesemer [21]

The relationship between intention recognition and communication can be seen best by considering intention recognition as a requisite for coordinating activity when communication is disabled. If communication is reliable and the agents are capable of fully and honestly and expressing their intent in an understandable fashion then intention recognition reduces to communication. There is no need for sophisticated inferencing, hypothesising, guessing, deducing, or evidence gathering. An agent simply asks another about its intent and the response provides all of the necessary information.

Without communication or other reasonably direct access to mental attitudes of others, to inform intention recognition it is necessary to rely on observations of the effects of their actions. It might seem that simply ensuring perfect, reliable communication of intention between agents offers a simple and efficient approach to intention recognition. Unfortunately there are many reasons why for almost all systems this solution is impractical. These reasons are useful to consider because they provide insights into constraints on the design of intention recognition that will guide the selection of better solutions.

Communication is unavailable In some systems it is not possible to implement direct communication between the agents. There are many reasons why this would be so. It may be bandwidth related, perhaps the agent is incapable of communicating.

Communication is unreliable The Byzantine Generals Problem [105] and its variants indicates the unreliability inherent in communication in certain situations. For many of these types of problems no communications based solution exists. Strategies for overcoming the difficulties of agent "mind-set synchronisation" exist but even without these logical design difficulties in distributed systems something as simple as the lack of a communication channel caused by a temporary hardware fault or transport delay can make communication unreliable.

Communication is uneconomic Communication is often very expensive. The lessons of distributed AI suggest that a balance between inter-process communication and processing needs to be achieved. If care is not taken the communication between distributed processes outweighs any advantage in distributing the problem in the first place. So in considering performance issues, particularly in real-time systems, balancing processing and communication is clearly an issue and understanding the balance is central to achieving good designs.

Communication is un-agent One of the advantages of agents is their inherent autonomy. By their very name they are expected to operate by themselves and require less assisting communication than some other non-agent solutions. That is not to suggest that agents shouldn't ever communicate—clearly communication is a very important part of the design of an agents. Overcoming problems by communicating more and more data to the agent when intuition suggests that the agent should be solving the problems for itself seems to violate the very notion of agency. Clearly care must be taken in balancing autonomy with the need for communicated assistance.

Communication is undesirable In competitive or combative situations communication is undesirable or might be deliberately deceptive. In commercial situations when security is of concern any communication must be treated cautiously. Similarly in military simulation where agent adopt adversarial roles any communication of intent to an opponent is unwise.

Communication is unrealistic The issue of 'realism' in agent systems becomes important when agents are used as computational models of real people. If agents are used as computational models of human cognition then there may be modelling fidelity requirements that dictate that the agents should not communicate if humans placed in a similar situation would not communicate. This is particularly relevant to military simulation where maintaining modelling validity with the real world is very important.

Communication is not understandable In heterogeneous agent systems it is likely that agents might speak different languages thus limiting the possibility for communication. Efforts to standardise communication languages [49, 50] continue but as yet it is likely that something as sophisticated as communication of intent is beyond the scope of these standards.

Viewing the content of observed actions as having a similar consequence with respect to intention recognition as communication suggests certain approaches for modelling intention recognition. This idea is elaborated in Chapter 3. These limitations refer specifically to the nature of communication but there are also impediments related to the content of the message.

Lack of Intention Not all agent models include an explicit representation of an intention, or even something that might pass for intention. Without an internal representation of intention the agent would need to construct some plausible representation of intention that modelled its behaviour for the exclusive purpose of communicating with others. This would seem to be highly impractical.

Intention is Complex Simply because an agent has an intention does not mean that it can communicate the important aspects of that intention to another. The specific implementation of intention within the agent would need to be communicated and that would depending upon the type of agent perhaps the sending a great deal of information about the goals, beliefs, plans, motivations or whatever the internal design of the agent dictated. In the extreme the entire internal state of the agent might need to be transmitted in order that intent can be correctly interpreted.

Intention is Contextual Intentions are often highly dependent on context. A high-level description of an agent's intent might not make sense without an understanding of the environment in which the agent exists. In order to communicate intent it might also be necessary to include a great deal of contextual information dealing with the surrounding environment.

Intention Recognition is Purposive An agent recognises the intent of another for a purpose. The depth of understanding of an intention that is required by an agent depends upon why knowledge of the intention is required. It might be that an agent needs only the broadest understanding of the intent. Simply knowing the general class of intent is adequate and details are unimportant. In order to obtain the appropriate level of detail a dialogue between agents would be necessary. In this light intention communication must be extended to a conversation between agents about the intention of one of them. This dramatically increases the complexity of intention communication from a simple broadcast by an agent to a negotiation about the nature of its intention.

With these limitations in mind it is easy to see that some form of intention recognition is likely to be needed to support coordinated or competitive activity in many domains. The following sections quickly references some of the more significant domains in which intention recognition is an identified requirement.

2.3.1 Human Intention Recognition

In everyday life human interaction requires ongoing and complex intention recognition. In competitive activities the need for intention recognition can be obvious but even in

simple conversations humans are predicting the future direction of the conversation and the reaction of the other persons by recognising the intent manifested in the conversation.

Any in depth analysis of human intention recognition is outside the scope of this thesis but a quick examinations of some of our everyday understanding of intention recognition will shed light on the remainder of the thesis. Often intention recognition is performed subliminally. Conversations unfold and subtle queues are reacted to without much conscious thought given to the underlying motives of the other.

Military focussed research into intention recognition is in the ares of automated support for the decision maker using data fusion and threat assessment tools [122, 190]. These tools are developed to process data from sensor networks and to assist the user in detecting and classifying targets or in determining the threat posed by unknown or hostile threats. The level of threat posed by a hostile contact is tied closely to inference of its intent. Endsley [44, 45] and others use a three level scale to classify the activities related to situation awareness. Level one situation awareness covers identification of friends and adversaries. Level two is the identification of the situation and the development of an understanding of the current activities of all participants. Level three situation awareness is the prediction of the future actions of others.

Most descriptions of intention recognition in practice posit a process that sees an expert sense the world, maintain hypotheses, and test and monitor for supporting evidence there is some more recent research that runs counter to this. The naturalistic decision making literature, adopts the ecological psychologists view of the fundamental importance of the environment and the very situated nature of humans and environment. NDM suggests that experts just *know*. There is little hypothesis evaluation or time spent on complex reasoning they simply recognise situations as a result of their experiences [194]. Norling and Heinze [126] have suggested a set of modifications to a rational agent architecture that allow for modelling of aspects the phenomena described by NDA. Zhang and Hill [192] have utilised similar templatised descriptions for modelling situation awareness in virtual worlds.

2.3.2 Agent Intention Recognition

Various approaches to implementing agent based recognition have been proposed [27, 92, 121, 91, 3]. These approaches to intention and plan recognition are grounded either in computer science or cognitive science. As such they inevitably adopt approaches that see complexity added to the internal computation of the agent. The standard is to observe the actions of others and to infer the internal states. This is natural enough from a cognitive modelling perspective in that it meshes with the sense and infer view of perception that dominates the psychology literature [177] and accords with the view of agents as autonomous. Even in multi-agent environments where the inference process must cater for uncertainty about the identities of the actors or their social structures the process is basically unchanged. Kaminka et. al. continue this thread of "observe and infer" in a multi-agent setting adopting an approach that is not too dissimilar from an earlier proposal by Tidhar and Sonenberg [162].

> *"Traditionally, agent modelling researchers have explored techniques in which two agents are involved. In such techniques one agent observes the actions of another agent and attempts to infer its unobservable state features, such as intent, goal, plan."*— Kaminka, Wendler and Ronen [90]

Agent modelling researchers seem reluctant to consider that either the environment or other agents might be specifically engineered to assist in the intention recognition process. This might be oversight, it might be due to a focus (to the exclusion of other aspects) of the agent, or it might be due to concerns about autonomy of the type expressed by d'Inverno and Luck.

> *"In multi-agent systems, the interactions between agents are the basis for usefully exploiting the capabilities of others. However, such a pragmatic approach has not been the concern of many researchers who instead focus on small areas of interaction and communication, and in particular on specialised forms of intention recognition and interpretation. In many existing models of interaction agents are not autonomous."*— d'Inverno and Luck [43]

It seems that in many applications there might be a trade-off between 'true' autonomy and cooperative or collaborative behaviours. Certainly their warning is valid though rather than adopt an approach that attempts to preserve autonomy outright this thesis will consider autonomy as yet another property of agency that might need to be traded-off against functionality to meet a given set of requirements.

There are cases where the agent whose intent is to be recognised is not amenable to engineering. This is certainly the case when the human intentions are to be recognised or when the agent lies outside of the design scope but there are many systems where the intending agent can be designed to enable intention recognition.

2.3.3 Agent Intention Recognition: A Case Study

A mature[14] example of an intention recognition capability operating in an multi-agent system is the one described theoretically by Rao and Murray [151], implemented as an extension to a commercial language by Busetta and Tidhar [19] and taken to a simple technology demonstrator by Tidhar, Heinze et. al. [168]. Bratman's philosophical insights that led to the BDI systems were intimately related to *practical reasoning*. Philosophers often characterise rational cognition as being composed of a practical component (what to do) and an epistemic component (what to believe) and it is here that BDI systems suffer certain limitations. Whilst BDI languages are inherently concerned with *beliefs* they offer no particular insights or support for the manner in which belief is created, maintained, or revised. Further there is no model of the environment and the way in which the agent perceives, observes, or recognises the world, or in the case of multi-agent systems, other agents. BDI systems that are well suited to practical reasoning are less well suited to epistemic reasoning. More fundamentally, there is no support for perception, observation,

[14]Mature in the sense that it is accompanied by an underlying theory and a set of programming language constructs in a commercial language.

sensing, and the other activities that characterise the ways in which agents must support their epistemic reasoning.

Within these limitations it is still clearly possible to construct a recognition of intention mechanism that can either (a) introspect over own plans to infer the currently executing plans of another [151] or (b) introspect over a predetermined set of possible plans [19]. In these cases and in most other systems exhibiting plan, goal and intention recognition some form of hypothesis tree is maintained with evidence being sought for accepting or rejecting particular options until ambiguity is resolved. Other systems integrate different technologies to handle the different aspects of agent cognition.

Systems for implementing resource bounded reasoning that define plans as structured descriptions for goal achievement have been used to implement complex human decision-making models. These systems use pre-specified plans as *recipes* [146] for the achievement of pre-specified ends. Previous research into plan recognition in these agent systems has required that the agent be provided with explicit representations of candidate plans [147, 151].

Work by Rao [147] tackled the reactive recognition problem by making simplifying assumptions about the nature of the environment: (a) the agent has perfect knowledge of the plans available to other agents; (b) the complete set of plans over which recognition is attempted remains small; (c) the agent has no memory of events that occur; and (d) the world is unchanging during the period of recognition. Whilst these assumptions result in serious constraints over the proposed model they do provide the basis for intention recognition that integrates well with the BDI model of Rao and Georgeff [149].

Work by Bussetta and Tidhar extended the functional capabilities of this type of plan recognition to remove the assumptions c and d [19] and to develop an operational extension to dMARS. A demonstration of the capability of this system to recognize mental states within a military simulation was constructed. These extensions to the computational BDI model added a model of memory and were able to incorporate some of the temporal aspects required for mental state recognition. The first two assumptions still limit the performance of system. In attempting to model human cognition it may be unrealistic (and perhaps impractical) to provide an agent with perfect knowledge of the plans of other agents. This is true in heterogeneous systems where agents must recognize the mental states of real humans, or of different varieties of agents that do not possess explicit representations of plans in a form understood by the agent attempting the recognition. As the complexity of the environment, agents, and their interactions increases, the set of plans over which recognition must be attempted will increase. Every agent within the system must be provided with a representation of the plans of all other agents that it is expected to recognize. Second order recognition (I recognize that she/he has recognized my plan) complicates this significantly and in complex domains will quickly become unwieldy.

Construction of the military simulation incorporating intention recognition revealed the problem that tends to beset much of the agent community involved in mental state recognition—perception. The BDI model [149] and the subsequent languages, PRS, dMARS, JAM, and JACK offer no inherent support for modelling perception. The agent model and languages provide an excellent framework for implementing practical reasoning but epistemic reasoning and the means by which knowledge of the world comes about is largely ignored. Given the heredity of BDI and its roots in Bratman's rational agency and its

focus on practical reasoning (to the detriment of epistemic reasoning) this is unsurprising but it leads to inevitable integration difficulties whenever a BDI agent must be integrated into a dynamic, complex environment. The solution described by Rao and implemented by Busetta and Tidhar addresses the agent issues by requiring the programmer to specify, design and code the "observations". Intention recognition is limited to an infrastructure for producing a set of hypothesised possible plans and removing or retaining them on the basis of these observations. It deals with the practical aspects but not the epistemic aspects. The complexity of modelling the perception required for these observations was still a significant impediment to the implementation of intention recognition.

These limitations aside, the demonstration provided a most valuable indication of the impact that intention recognition can have on military simulation by demonstrating the capacity for aircraft to avoid feinting or deliberately deceptive behaviour. The solution is in keeping with the conceptualisation behind the BDI model. It could be argued with some justification that a strength of the agent paradigm lies in the higher level of abstraction it offers. In the case of dMARS and JACK their relevant developers would both argue that the abstract specification of their reasoning is a strength and not a limitation. That this makes some tasks of modelling perception difficult is a limitation of the environments into which agents are placed and not the agents themselves.

2.4 Situated Cognition and Ecological Psychology

The situated cognition and ecological psychology literature describes the strong links that tie perception, action and cognition to the environment. The status of situated cognition and ecological psychology is still hotly debated and the arguments for against often reduce to the behavioural psychology versus cognitive psychology battle that has been fought for more than a century. Whether or not the claims of Gibson, Clancey and other advocates are believed often distills into fundamental ontological and epistemological stances[15].

> *"Especially in the philosophical literature, the claim that vision involves inference typically derives from certain general considerations about the nature of empirically gained knowledge. The basic idea running through most versions of this criterion is that there is an epistemologically important difference between what we can "really" ("simply", "directly", or "immediately") see and some of the other things that we can find out about by means of vision."*—Robert Schwartz [158]

In the real world, filled with real people there is little doubt that psychologists will debate the issues for many years to come but in virtual worlds (if the temptations of good

[15]There is an analogy between the views that Schwartz contrasts here and the broad separation of the agent-ontology community into two camps. There are those that argue that ontologies should be global—suitable for all agents. This makes a strong statement that there can be an objective reality to which agents commit. This reduces the chance for agents to develop local, subjective views of their individual realities. To be fair the agent ontology community is proposing standards for strongly pragmatic reasons rather than adherence to some philosophical standpoint.

old fashioned AI are resisted and a pragmatic approach adopted) there are insights from situated cognition and ecological psychology for the construction of agent systems.

For example, failure to appreciate the non-symbolic nature of sub-conscious aspects of perception has led to difficulties in modelling at the interfaces between agents and environments[16]. Clancey states this in 'Situated Cognition':

> *"Conventionally, the term symbolic processing is used to refer to what people and expert systems do, misrepresenting the subconscious aspect of conceptualisation and thus superficially equating reasoning with the calculus of descriptive cognitive models."*—William Clancey [30]

Further, Clancey critically assesses the close links between activity and perception that are ignored by the philosophical camps that ignore or oppose situated cognition.

> *"Fodor and Pylyshyn do not acknowledge the distinction we experience between direct perception ("I felt it and knew") and inference ("I noted, checked, and confirmed"). Instead they say, "Object recognition, for example, is a perceptual process par excellence, and it appears to be cognitively penetrable through and through. But object recognition is not a unitary phenomenon... The elevation of recognition to a primary phenomenon in psychology, existing independently of purposes and action highlights the mistake: perceiving is not primarily identification but functional differentiation, part of an activity."*—William Clancey [30]

The simple message of ecological psychology and situated cognition is that the environment and the agent are more intricately entwined particularly with respect to perception—the primary interface between the two. The environment should be considered as something that changes in response to the state of the agent just as the agent responds to changes in the state of the environment. This perspective has been largely ignored in the agent literature, there are several application domains that have (often unwittingly) adopted the approach for sound engineering reasons[17]. This has implications for modelling of environments as well as the modelling of agents as highlighted by Turvey and Shaw.

> *"A change of pace or a change of location can mean that a brink in the ground now affords leaping over whereas at an earlier pace or location it did not. [...] The environment-for-the-organism is dynamic and action-oriented while the environment-in-itself, that which has been the target of most modelling in the latter decades of the present century, is fixed and neutral with respect to the organism and its actions..."*—Turvey and Shaw [173]

With intention recognition in mind the ecological psychology literature of most interest is that which deals with perception of the environment. Traditional theories of perception, the theory of perceptions as hypotheses, or 'indirect perception', claims that it is not

[16]Chalmers, French and Hofstadter warn of a related problem when the state that AI research has failed to appreciate the very tight coupling that exists between high-level perception and cognition [26].

[17]Several examples of this are given in Chapter 3.

possible to be directly aware of the actual physical world [177]. Human vision adds to the incoming sensory data by inferring higher order properties and structures. To make sense of the world requires that the sensory data is somehow elaborated and abstracted into models, schema, hypotheses, pictures, or mental images. The process of elaborating and abstracting sensory information depends on memory, experience, situation, emotional and physical state and all of the mental aspects that collectively determine how the perception of objects occurs.

J. J. Gibson countered this theory with a radical proposition that resulted from his study of ecological psychology. Gibson proposed that animals and environments have co-evolved:

> *"The words 'animal' and 'environment' make an inseparable pair. Each term implies the other. No animal could exist without an environment surrounding it. Equally, though not so obvious, an environment implies an animal (or at least an organism) to be surrounded."*—Gibson The Ecological Approach to Visual Perception

This co-evolution has attuned vision directly to the higher order structures[18] that exist in the environment and they are directly perceived (See Figure 3.2 and refer to Section 3.2 for more detail). Gibson's theory is at its most controversial when the concept of affordances is introduced. Affordances are properties of objects in the environment that offer action possibilities to the observer, a chair for example has a 'sit upon' affordance. Affordances, Gibson claims, are directly perceivable in the environment in precisely the same way as color or size.

The notion that objects are envisioned in terms of the action possibilities that they afford was extended by Norman [127, 128] by incorporating affordances into design. The idea that agent systems might be similarly treated and designed in terms of their affordance is explored in later sections of this thesis.

2.5 Intelligent Agents and Artificial Intelligence

There are two distinct but related views of artificial intelligence. One is concerned with the construction of intelligent artifacts that serve useful purposes and is therefore primarily an engineering endeavour. The second is the investigation of the computational modelling of human cognition—and is therefore primarily scientific in nature. The latter has led to insights into human intelligence as well as passing useful theories and ideas to the engineering community to adopt and make mainstream. Recently AI has been transformed (at least partially) by flourishing agents research that shares these two facets. Some agents research is associated with applying, studying, modelling, or learning from human behaviour. Others are less concerned with human behaviour and more concerned

[18]Gibson referred to these as invariants. There are two types of invariants: transformational invariants that govern the manner in which objects change as they move; and structural invariants that relate relationships between objects regardless of their locations. An example of this is the property that the ratio of an object's height to the distance between the base and the horizon remains constant for all locations of the viewer.

with engineering of artifacts that display behaviour that might be classified as intelligent—even if it shares little in common with human behaviour.

Throughout the last decades there has been a shift from monolithic general problem solvers and large expert and rule-based systems toward computation that is smaller, social, embedded, mobile, and networked. In making this shift AI has been transformed from the study of human intelligence (singular, isolated, and disembodied) to the study of human intelligence (social, plural, and situated). A side-effect of this is that the boundaries of agents (the points at which they mesh with the environments they inhabit) now pose a number of problems to those who would design and develop agent systems.

The importance of the environment in conditioning the design of agents has been recognised:

> *"When one thinks about building intelligent agents, it quickly becomes obvious that the task environment in which the agent will operate is a primary determinant of the appropriate design."*—Michael Jordan and Stuart Russell [88][19]

Though recognising the importance of the environment in conditioning the design of the agent few (if any) acknowledge that the agent and environment are coupled for the purposes of design and that the environment might be specifically engineered to meet the requirements of the agent.

The internet, as the largest example of a virtual world, is one environment where steps have been taken toward *agent-friendliness*. The early life of the web was based around HTML and structured pages in ways suitable for access by browsers that presented information to users via a display. XML and other approaches to structuring knowledge on the web provide support for agents, bots, search-engines, b2b protocols etc. in traversing, finding, learning and navigating. For more information on the importance of labelled environments to agent design see Section 3.3).

2.5.1 Perception and Related Issues

In order to act within an environment an agent must be provide with sensors that allow appropriate perception. In the case of robots these sensors are normally machine vision systems, natural language processing and understanding systems, haptic systems or some other customised sensory apparatus that provides an interface to the real world. In virtual worlds the sensors might not need to be quite so complex, virtual worlds tend to be simpler, discrete, and designed. Even so the interface between agent and environment requires careful design.

> *"The data input and output subsystem must be carefully matched to the task performed—adequate sensors and effectors must be provided, and go a long way in bringing about the agent's functionality. Attempts to skimp on the I/O and compensate for it with more computation are normally ill fated."*—Brustoloni [16]

[19] See also the quote by O'Dell in Chapter 1.

Perception is the means by which an agent senses and makes sense of its environment. Exactly which processes should be included into the broad definition of perception is somewhat controversial. Indeed the means by which human perception integrates with cognition is hotly debated making the task of creating an artificial intelligence that resembles human intelligence even more difficult.

Perception is continually included in the list of important agent concepts (see Section 2.1). It is clearly also a very important component of intention recognition. Intention recognition is a process that straddles the boundary between cognition and perception. To recognise the intent of another seems to imply some high-level cognitive processing and yet there are times when it seems intuitive, and hence inaccessible to introspection [194].

It was noted in Section 2.1 that agents typically function with more abstract data than other software. Converting from the low-level data available in the environment to the knowledge-level structures required by the agent is appropriately handled by perception. In this role perception acts as a bridge over the abstraction gap [77].

In addition to sensing the environment, the social nature of agents means that they communicate with other agents. Even if this communication is mediated by the environment the fact that agents all function with knowledge level concepts means that translation at the knowledge level and not abstraction is the important issue [84]. Thus perception might also be viewed as the mechanism by which knowledge level concepts are translated from one agent to another. Or, adopting the language of the ontologies literature: perception is the process of mapping the data in the environment into the more abstract representations of the agent ontology; and perception is the process of mapping from one agent ontology to another.

The machine vision system of a robot might perceive a colored pole based upon its edge detection algorithms and determine that it has a color property of being red and yellow. Within the robot there will be some mapping from the raster scans of its vision system into objects, which when assigned meanings can be reasoned about. Each object must be detected, isolated, tracked, identified, and interpreted. This process is highly complex and machine vision systems are amongst the most sophisticated robotic software applications developed. Machine vision is really only necessary in dynamic real worlds. If the world in which a robot operates is fairly static then a vision system may not be necessary. Manufacturing robots daily build cars with only location and contact sensors as guide. The regularity of their world means that they don't require the sophistication of vision. A manufacturing robot working on a production line repetitively performing the some welding task can be considered to be living in a highly designed environment. Careful design of the environment ensures that the robot is not presented with situations outside the limits of its simple sensors. This is a very good example of how environmental design mitigates against agent perception complexity.

The classic robotics literature challenge for a robot is to traverse a series of offices emptying bins (and perhaps vacuuming the carpets.). It turns out that even this seemingly simple task is made difficult by the uncertainty about the location of furniture, people, rubbish bins etc. To navigate around offices populated by people requires more sophisticated sensors. Perhaps, as Brooks suggests, no sophisticated internal representations of the environment are required but still there is a strong requirement to be able to sense the complex dynamic world [13]. Throughout the robot world there are examples that

reinforce the view that there are trade-offs between the design of the environment and the perception systems that drive the robots. In complex systems where little design control can be exerted over the environment the demands on robot perception are high. Robocup [98] is a very good example of a system in which, even though some design control is exerted over the environment, the robots still require very sophisticated perceptual apparatus.

Just as there are similarities between agent communication and intention recognition so are there similarities with machine vision. The lessons from machine vision for intention recognition are obvious. Design the environment and perception is simplified. Just as colored poles surround a Robocup field to help the robots localise imagine the prospect of robots that changed color from moment to moment as their intention changed. Though this is really just another form of communication, in virtual worlds the prospect of engineering perceivable artifacts into every aspect of the environment offers different possibilities for assisting intention recognition.

If there is little or no design control over the elements of the system other than the agent, then there is clearly a need to provide the agent with the facility to deal with the unconstrained complexities of a dynamic world. In terms of intention recognition this might require a module akin to a robotic vision system that takes sensory data from the world and performs some complex mapping onto symbols that are in the form necessary for agent reasoning. The robotic vision system typically processes video images into named, attributed objects. Depending upon the specific domain intention recognition might process object-attribute information and convert it into some symbolic representation of intention. The sensory data and the resultant information is different but the process is conceptually similar.

Difficulties are also apparent in defining the boundary of perception and cognition. Chalmers at. al. deal with this via their theory of *high level perception* that provides a representation of the processing that deals with the abstraction, fusion, and recognition based perceptual tasks.

> *"High-level perception—the process of making sense of complex data at an abstract, conceptual level—is fundamental to human cognition. Through high-level perception, chaotic environmental stimuli are organised into the mental representations that are used throughout cognitive processing. Much work in traditional artificial intelligence has ignored the process of high-level perception, by starting with hand coded representations."*— Chalmers, French and Hofstadter [26]

Elsewhere too the limitations caused by the symbol binding problem for purely symbolic reasoning systems has addressed with the obvious conclusion that human (and possibly agents) require some form of sub-symbolic processing occurring in the interface between the environment and high-level cognitive activity.

> *"A related problem for purely symbolic approaches is that sensory information about the physical world is usually thought of as numerical. Thus there must be a layer of non-symbolic computation between the real world and the realm of pure symbols. Neither the theory or the practice of symbolic AI argues against the existence of such*

a layer but its existence does open up the possibility that some substantial part of cognition occurs therein without ever reaching the symbolic level."—Michael Jordan and Stuart Russell [88]

2.5.2 Explicit Representation

This thesis will argue strongly that an explicit and abstract representation of *intention* is a necessary and useful precursor to implementing intention recognition because it simplifies the process of design. In taking this stance this thesis is aligned with those agent and AI researchers who advocate agent systems that explicitly represent particular aspects of intelligence:

"It is not clear how an agent might undertake intention recognition without some explicit representation of intention."—Mike Wooldridge Presentation to the Melbourne University Agent Lab, 2000

Yet Rodney Brooks and other criticise mainstream AI for abstracting away important aspects of intelligence and overly simplifying the problem:

"The only input to most AI programs is a restricted set of simple assertions deduced from the real data by humans. The problems of recognition, spatial understanding, dealing with sensor noise, partial models etc. are all relegated to the realm of input black boxes."—Rodney Brooks [13]

This is a manifestation, albeit in a modified form, of the connectionist versus symbolicist argument that has split AI in the past [120]. Brooks prefers systems that have no explicit representation of the environment (much less intention) but still manage to generate intelligent behaviour.

"When we examine very simple level intelligence we find that explicit representations and models of the world simply get in the way. It turns out to be better to use the world as its own model. "—Rodney Brooks [13]

Reconciling these two seemingly different views is unnecessary because there are fundamental differences in the types of activities with which Brooks and others are concerned and those addressed by this thesis. The ultimate aim of this thesis is to contribute to the general understanding of the means by which intelligent behaviour might be engineered, a goal not dissimilar from Brooks and his robot development but Brooks takes the view that:

"We must iteratively build up the capabilities of intelligent systems, having complete systems at each step of the way."—Rodney Brooks [13]

Furthermore Brooks is primarily interested in systems that function in the real world (i.e. robots) whereas the predominance of agents operate in virtual worlds where the environment is itself simplified and abstract.

This thesis is concerned primarily with *modelling* intelligent behaviour. Taking a software engineering stance almost presupposes simplification and abstraction of the problem to support design. The task of software engineering is to manage the complexity of software systems, and abstraction is one of the tools available to the engineer.

So what Brooks sees as a deficit, simplifying the problem by abstracting the details, is one of the important messages of this thesis and so this thesis is about modelling and design rather than implementation. Indeed, Brooks' robots are not *designed* in software engineering sense but are constructed over time, evolving their intelligence iteratively as a *mechanical whole* situated in an environment.

Certainly this thesis allows for the modelling of systems where the *intending agent* has no accessible internal representation of intent. But if there is no representation of intention in the *recognising agent* then then there is surely no intention recognition. At best there is the *appearance* of intention recognition but at design time, if intention recognition is required then those intentions must be described.

So the argument for an explicit representation is threefold:

1. *Explicit representations support design.* Abstract simplified and explicit representations are useful in modelling and designing agent systems, even if the implementation fails to maintain those representations. Without explicit representations to guide the design the *intelligence* is an emergent property of the system. In other words, making appropriate design choices about a system to implement intelligent behaviours requires explicit representations of the important parameters that influence the design.

2. Intention recognition is too sophisticated a behaviour to arise as an emergent property of a system. Without an explicit representation of intention in the *recognising agent* the task of effectively generating intention recognition is problematic at best. Brooks' robots do exhibit intelligent behaviour, but that behaviour is ascribed by the observer and does not exist as an accessible description inside the robot. Further more Brooks himself classifies the behaviours with which he is concerned as "simple", a label which is inappropriate for intention recognition. Ascribing *intent* to a robot is plausible but it seems less likely that that an observer would ascribe *intention recognition* to a computational entity.

3. In virtual worlds the environment can itself sustain explicit representations of simplified abstract concepts that can support intelligent behaviours. Brooks' point has validity for the construction of robots that must operate in the continuous complexity of the real world but is less valid for the construction of AI that live in virtual environments. In virtual worlds the environment is (almost always) simpler than the real world and can be augmented, abstracted, and adapted to better suit the agent and to support the generation of the abstract simple concepts with which the agent reasons.

2.5.3 Hybrid Systems

Robotics has, of necessity, patched dissimilar technologies together for decades. From a purists perspective robotics might seem to be less about a search for good old fashioned AI and more about tinkering with electrical engineering but the lessons that have been learned in robotics can inform some aspects of agent development. The emerging view of agents as embodied, situated, and social shares more in common with robots than it does with the monolithic rule and knowledge based systems that once characterised AI. The differences between the types of processes normally categorised as perception and cognition suggest that (wherever the boundaries are drawn) different implementation technologies are likely to be required if it is necessary to maintain links to human intelligence. Minsky's Society of Mind [119] develops a view of intelligence as a grouping of coordinating agents that together provide intelligence.

The increasing demands being placed upon intelligent agents as the technologies mature seems to suggest that future AI (or at least parts of it) will require the assembly of hybrid systems from a variety of technologies. It was true when Minksy wrote of it in 1991 and it is true now.

> *"In the 1960s and 1970s, students frequently asked, "Which kind of representation is best," and I usually replied that we'd need more research before answering that. But now I would give a different reply: "To solve really hard problems, we'll have to use several different representations." [...] we must develop systems that combine the expressiveness and procedural versatility of symbolic systems with the fuzziness and adaptiveness of connectionist representations. "*—Marvin Minsky [120]

A similar sentiment is expressed by Clancey in highlighting the different character of activities generally classified as intelligent.

> *"Put another way, by ignoring consciousness, researchers have swept under the carpet those aspects of intelligence that cognitive modelling fails to explain. Reasoning is inherently a conscious activity; what occurs within cycles, subconsciously, is not more direct manipulation, but a process of a distinctly different character, involving different modalities, not just verbal, which allows describing to occur and gives it the power it has to change our behaviour."*—William Clancey [30]

The reliance on a single technology or unified approach seems ill conceived. The solution to developing computational representations of intelligent behaviours is to develop a variety of approaches that allow technologies to be adopted on an *as needs* basis.

In a very practical sense this view is being adopted by developers of games that use agent technologies.

> *"Of the concepts I've presented in this article, there are three features from Black and White*[20] *that I expect will become increasingly commonplace in game AI in the coming years. First, agents' minds will come to include both symbolic and connectionist representations, happily coexisting in one unified architecture."*—Richard Evans [46]

[20]Black and White is a computer game released by Lionhead Studios that uses BDI architectures for its characters.

2.5.4 Pattern Matching and CLARET

Pattern matching techniques are used in Chapter 7 to implement part of the intention recognition system of an agent. The particular algorithm, CLARET [137], has been used in a number of domains and is strongly suggested for this application by virtue of its capacity to deal with spatio-temporal relationships and to produce successful matches as soon as the choices are disambiguated. This allows the results to be incorporated into an agent with the capacity recognise the behaviour of another prior to its completion. CLARET structures its examples hierarchically allowing advanced applications to build sets of nested behaviours to be recognised and announce the successful discovery of these as recognition progresses. This has definite analogies to intention recognition systems that maintain hypotheses in the form of sets of plan instances. The difference with CLARET is that the matching is done of the basis of learned examples that are partitioned automatically by the algorithm. In the CLARET algorithm an unknown segmented and labelled trajectory case is presented to the system together with examples of known trajectories using a simple polygonal approximation technique. Relationships between trajectory segments are extracted and parameters defining their relationships are calculated. Relational rules are generated that explicitly depict relationships between states. Matching techniques relate task descriptions to the available data. A parser is then used for conversion of events matched using these rules into descriptions that are consistent with the syntax of the events and their relationships. The system uses inexact (approximate) matching and a Bayesian probability network to negotiate between alternative hypothesis, and thus the rules are transferable to other airports and under variable conditions.

2.6 Software Engineering

The previous section stressed the importance of environments to agents and provided examples of the manner in which careful design to reduce the load on an agent's perception related tasks. The design of agent software should draw on the experiences of the software engineering body of practice. Too often agent research forgets or ignores the lessons of software engineering with the result that solutions are tailored to narrow or toy domains and are consequently difficult to reapply.

There are a number of fertile areas in the state of the practice of software engineering. With the aim of this thesis being a means of modelling intention recognition in agent systems some of the more obvious topics include: design patterns as an approach to reuse; software modelling languages such as the UML; and proposed agent oriented software engineering methodologies. Each of these areas is covered in the following sections.

2.6.1 Design Patterns

Design patterns are abstractions from instances of design forms that reoccur in specific software development. Typically they are "geared toward solving problems in design" [154] and therefore simplifying the implementation.

A high-level definition of a pattern is provided by the Patterns Web Page:

> *"A pattern is a named nugget of insight that conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns. Patterns are usually concerned with some kind of architecture or organization of constituent parts to produce a greater whole."*— [134]

A working definition that gives greater insight into a pattern's composition is:

> *"Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves."*—Gabriel [52]

Once a collection of related patterns is available and experience with their use is achieved it is possible to develop a Pattern Language. Again from the Patterns Home Page [134] a 'Pattern Language' is

> *"a collection of patterns forming a vocabulary for understanding and communicating ideas. Such a collection may be skillfully woven together into a cohesive 'whole' that reveals the inherent structures and relationships of its constituent parts toward fulfilling a shared objective. If a pattern is a recurring solution to a problem in a context given by some forces, then a pattern language is a collective of such solutions which, at every level of scale, work together to resolve a complex problem into an orderly solution according to a pre-defined goal."*—Patterns Web Page [134]

With respect to productivity, patterns are one of the more significant ideas to be introduced to software engineering. Specifically they support the cataloguing of knowledge about successfully employed solutions to well understood problems. In doing so, they have the capacity to *short-circuit* the search for answers to a software design question and present the engineer with range of options suitable for the task. A set of agent software design patterns for modelling intention recognition is provided in Chapter 6.

Cognitive patterns are an extension of standard software design patterns that allow modelling of human problem solving in a software development environment.

> *"As systems become more complex, the human limitations to comprehending system requirements become more evident. Since we cannot develop appropriate solutions if we do not understand the problem, human understanding is the key ingredient. Cognitive Patterns addresses this central issue by providing techniques for system specification that are based on our human facility for thinking and reasoning. As such it does not model system requirements in terms of programming languages and platforms. Instead, it models the way reality is understood by people."*—James Odell [55]

The use of cognitive patterns has not been widespread in practice, but the concept of providing a modelling framework based on human reasoning that is useful for system specification is powerful. The look and feel of the cognitive patterns literature [55] is adopted for expressing the architectures of the approaches to intention recognition proposed in Chapter 5.

2.6.1.1 Agent Patterns

A clear and present need for documented agent patterns exists [100]. Patterns may manifest initially as *sample code* in a variety of agent languages, or as unstructured descriptions of solutions to problems, but as time passes there will be increasing demand for the tools and practices that have proven useful in other software paradigms to be adapted or adopted for use developing agent systems. Patterns have proved to be a valuable contribution to productivity in software development generally. The importance of architectural design in agent systems (indeed any distributed system) suggests that there is every reason to believe that *agent patterns* will result in similar advantages.

Design patterns have been under-utilised for agent systems. Experience with agent software patterns is limited by the recent advent of agency as a software paradigm and the scarcity of documentation describing agent systems at the architectural level. Some notable exceptions are the work by Kendall et. al [95], Deugo et. al. [40], Meira et. al. [34], Lind [108] and Findler et. al. [111]. In each of these cases the authors present agent designs in the form of a patterns catalog and although the approach to documentation differs in each case they are all generally consistent with the mainstream software engineering patterns literature.

Just as object oriented patterns describe either relationships between objects or the internal structure of objects so to can agent patterns describe relationships between agents and their environments or the internal structural details of the agent. Design patterns for agent systems will include particular agent architectures, cognitive models, and, in the case of multi-agent systems, teams and social structures. Chapter 6 deals with the provision of a set of design patterns concerned with the modelling of a particular functionality within an agent system—intention recognition.

There are some fundamental differences between agent oriented systems and object-oriented systems that influence the presentation and content of patterns. Foremost amongst these is the relatively higher level of abstraction required by an agent pattern. Many object-oriented patterns are specified in terms of concrete classes. In some cases the patterns are detailed enough to approach source-code templates and are described with sample code. Agent system developers will benefit more from architectural patterns that capture the interactions between agents and environments and among the agents as they collaborate. Detailed *source-code* level patterns (referred to in the literature as programming idioms) that specify the designs of internal components of the agent are useful but will have limited scope for reuse due to the large differences between agent theories, architectures and languages. Design patterns strive for language independence but the fact that agent languages exhibit more diversity than the comparatively uniform object-oriented languages has implications for the provision of agent design patterns. This, and the more abstract nature of agents, requires more abstract design patterns than are observed in object-oriented analysis and design. Design level descriptions, critically documented and analysed, are a more valuable resource for the agent system developer than sample source code [21].

[21] If agent oriented software matures to the same level as OO and there is a rationalisation of languages with a standard set of concepts then source code will become increasingly useful.

2.6.2 The UML and the AUML

The UML is an appropriate tool for modelling agent systems. It is extensible and supports much of the modelling that is required. Where it is deficient there are efforts underway to address the issues. The deficiencies are small compared to the advantages it offers.

Throughout this thesis a number of extensions, modifications, or adaptations of the UML were introduced to cater for specific agent development modelling requirements. This Chapter gathers those together not to provide an authoritative or complete agent modelling language but at least a thoughtfully composed set that have proved useful for the purposes of the analysis and design methodologies.

Attempts have been made to maintain compatibility with the general development of the UML and, in particular, the AUML.

The UML and associated methodologies [102] are rapidly becoming industry standards for the development of object oriented systems. Unsurprisingly, this popularity has led to the development of agent oriented extensions. Foremost of these is the Agent UML (AUML) [132]. Experiences with the development of military simulation has echoed these wider international trends. In systems such as SWARMM and BattleModel only a small percentage of the development effort is agent related. Much of the system design effort is focused on aspects that have little or no relationship to agency and can be tackled with traditional software engineering techniques. The Rational Unified Process, the UML, and standard software engineering tools have proved the worth in the development of the simulation infrastructure and the OO aspects of the system. Whilst techniques for modelling, designing, and specifying the agents are immature preliminary experience with the UML for requirements specification [73] and design [136] has proved promising. There is significant management, training, and infrastructure advantage in maintaining a single tool set for agent and object oriented components of a system.

2.6.3 Agent Oriented Software Engineering

The agent oriented software engineering is almost as diverse as that dealing with other aspects of agency. Broadly there are two camps: those attempting to reapply (perhaps with substantial modification) successful object oriented approaches [17, 131, 132, 73, 136, 38]; and those attempting to develop agent oriented techniques (often these are associated with a particular agent theory, architecture or language) [33, 94, 47, 8, 185, 182]. Neither group dominates the literature and experiences with large scale agent development are so rare as to make empirical evidence statistically insignificant. It seems likely that the future will see some compromise that satisfies the needs of most. Juan, Sterling and Winikoff [89] have proposed a modelling approach that allows a core agent model to be extended to meet the needs of a variety of users.

There are several good summaries of the agent oriented software engineering literature [189, 187] that provide detail in areas outside the scope of this thesis.

Software engineering has been reported by several researchers as one of the significant remaining challenges facing the agent community [123] and if the lessons from the OO

community are valued it seems that many of these challenges will not be addressed until more broadly based experience with the construction of agent systems is published. In the interim those constructing agent system in an industrial setting make use of the tools that they have at their disposal. It is precisely because of this that the UML (and its agent oriented sibling) the AUML receive so much attention in agent software engineering circles. The agent based software engineering workshops at ICSE (International Conference on Software Engineering) or AAMAS (The International Conference on Autonomous Agents and Multi Agent Systems) see many examples of agent extensions to UML.

In order to explore the current status of agent oriented software engineering and to move some of the way toward predicting its future, comparisons will be drawn with the development of object oriented technologies and methodologies. Whilst it is by no means clear that these comparisons are valid they at least off some insights into some of the issues currently waiting to be addressed by agent practitioners and some possibilities for the future.

This thesis ascribes to the view expressed by Jennings [85] that agent based software engineering will become mainstream. This is due to it being the "natural next step" and being an appropriate model for developing the open highly networked systems that are starting to dominate the software engineering landscape. There is a third point that Jennings does not make, though it is related his first. Not only are agent the natural next step, but they are just plain natural. They provide software engineers with a useful set of concepts for managing system complexity but even more interestingly they provide the non-expert with some insights into complex system development in a way that object systems do not. Some researchers have reported the benefit of exactly this property of agency [74]. If the potential of this is realised it may lead to a suite of technologies that help to bridge the gap between customer and developer.

Object oriented languages, ADA, JAVA, C++, SmallTalk, etc. have much the same object oriented language features. This allows design techniques to be applicable to more than one languages. Indeed it is considered good practice in the OO community to produce designs that are as language-independent as possible. Unfortunately for the agent community the diversity of agent languages makes language independent designs an impossible goal. Efforts can be taken to keep the software engineering process language independent as long as possible but a point will be reached, much earlier than in OO, where a commitment to a specific delivery architecture will greatly influence the design. Indeed, it is not altogether clear that this agent diversity does not impinge upon the analysis phase suggesting that the way we analyse a system might be, rightly, influenced by the agent langauge.

2.7 Ontologies

From philosophy comes a definition of ontology as:

> *"that department of the science of metaphysics which investigates and explains the nature and essential properties and relations of all beings, as such, or the principles and causes of being."*—Websters Dictionary [23]

Philosophers regard ontology as the science that seeks to understand what it is that actually exists, what relationships hold between that which exists and what it is that cause existence.

Unsurprisingly philosophical views of ontology differ widely and there are often ambiguities and overlaps between what is regarded as ontology and what is regarded as epistemology. The distinction being broadly that ontology is concerned with what *is* and epistemology concerned with what *is known.* In the detail of particular views lies the very essence of much of philosophy and how those views have shaped science over the millennia.

Ontologies as they pertain to this thesis will define the knowledge level concepts that are useful in representing agent knowledge. Accepted definitions as used by AI practitioners where ontologies refer to theories about the properties and relationships that exist between entities in a system.

> *"An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory."*—Gruber [63]

Ontologies are useful for information exchange and inter-agent communication [49] and for assisting in the specification of information systems [67], and for broader knowledge management activities that are not necessarily software related [5]. Uschold [175] describes three purposes to which ontologies are put: communication between people, inter-operability among systems, and for systems engineering (and it is the system specification aspect of ontologies with which this thesis is concerned). In use, practitioners do not require an ontological commitment by an agent actually result in associated symbolic reasoning. The agent must implement an interface that allows it to communicate and interact with other agents as if it does.

> *"We use common ontologies to describe ontological commitments for a set of agents so that they can communicate about a domain of discourse without necessarily operating on a globally shared theory. We say that an agent commits to an ontology if its observable actions are consistent with the definitions in the ontology. The idea of ontological commitments is based on the Knowledge-Level perspective (Newell, 1982) . The Knowledge Level is a level of description of the knowledge of an agent that is independent of the symbol-level representation used internally by the agent. Knowledge is attributed to agents by observing their actions; an agent"knows" something if it*

> *acts as if it had the information and is acting rationally to achieve its goals. The "actions" of agents—including knowledge base servers and knowledge-based systems—can be seen through a tell and ask functional interface (Levesque, 1984) , where a client interacts with an agent by making logical assertions (tell), and posing queries (ask)."*"—Gruber [63]

This echoes the distinction that has been made between ascription and description for agency and intentionality. In much the same way Gruber acknowledges that ontologies do not require an internal representation of the knowledge concepts, only that their behaviour is consistent with the ascription of those knowledge concepts.

But Gruber also makes the point that:

> *"as a software engineering construct ontologies play the role of a coupling interface between knowledge bases..."*—Gruber [62]

This highlights the important role that ontologies play in the specification of system interfaces and that they can provide system-level descriptions of the design. Gruber goes on to provide design criteria for the assessment of ontologies. Clarity, coherence, extendability, minimal encoding bias, minimal ontological commitment. Many agent developers use ontologies as knowledge-level descriptions of the important concepts manipulated by the agent [54, 12]. In this sense they are design-level descriptions of agent functioning.

The proliferation of research into ontologies related to agent system development is strongly related to the abstract nature of agent reasoning and the desire to create computational entities that reason and communicate with higher orders of data. Wiederhold makes the point that:

> *"Data from multiple sources will often not match in terms of naming, scope, granularity of abstraction, temporal basis, or domain definition."*—Weiderhold [180]

and that this mismatch is exacerbated as reasoning is conducted at more abstract levels:

> *"... the information required to initiate action will be hidden in ever-larger volumes of detail, scrollable on ever larger screens, in ever smaller fonts. In essence the gap between data and information will be wider than it is now."*"—Weiderhold [180]

There are two important aspect of Wiederhold's discussion of ontologies. First is that agents often commit to different ontologies and that a process of translating ontologies is required. Second is that the knowledge concepts that the user or agent requires, and that should be represented in the ontology, are not always available and that some process for translating from the system level data into the higher order concepts represented in the ontology is required.

Regardless of the explicitness of the agent ontology there is a need for methods to assist the engineer in constructing ontologies. Several authors have proposed approaches

for addressing a clear need for a systematic approach to the development of ontologies. Uschold has the aim of developing a unified methodological approach to building ontologies and comments that:

> *"The main barrier to the production of such a coherent unified framework embracing all of these techniques and methods for building ontologies is that there is no clear indication of how general the individual techniques and methods reported to date are."*—Uschold [175]

Mariano [58] and others have also indicated the need for systematic approaches to the construction of ontologies and Menzies et. al. describe automated approaches for evaluating the utility of ontologies[118]. Clearly there is a need for further insights into the design and use of ontologies in agent systems.

2.8 Summary

The background literature has indicated that the following areas are of primary importance to a consideration of models of intention recognition:

1. existing approaches to intention recognition focus on adding functionality to the agent and tend to ignore other alternatives;
2. although perception is widely considered a fundamental property of agency there has been relatively little work done to more accurately specify the concept;
3. techniques are required to design ontologies for multi-agent system;
4. many systems that might require intelligent agents with the capacity for intention recognition will be constrained by performance requirements;
5. a range of solutions, particularly those which cater flexibly for hybrid technologies, are required.

To address these issues in the context of the aim this thesis draws together several key research threads:

1. the work of Rao and Murray [151], and later Rao [147, 151], Rao and Georgeff [149] and Busetta and Tidhar [19] that resulted in an agent based implementation of intention recognition that uses a reactive recogniser as a component of agent reasoning. This research provides a strong example of the traditional approach to intention recognition applied to a relevant domain. Similar examples are provided by Kaminka and Pynadath [144] and others [60];
2. the situated cognition literature particularly Clancey [30] and the ecological psychology of Gibson [57]. In particular Gibson's theories of ecological visual perception, and the idea that higher order structures are directly accessible in the environment [56] and the further utilisation of those ideas for design proposed by Norman et. al. [128];

3. labelled environments and the idea of designing the environment to cater for agent perception. A recurring theme in games [81, 46], the design of web pages [163, 28], and even RoboCup [98];

4. the ontology literature, particularly Gruber [63, 64] and Wiederhold [180];

5. the intentional stance and the work on ascribed intentionality of Dennett [39] and Bratman [10];

6. the work of Pearce [137] and Pearce, Caelli, and Goss [138] in pattern matching, particularly as it pertains to the classification and recognition of complex human behaviour [139];

7. the software engineering literature that pertains to patterns [53, 18] and the related literature that has emerged from the KADS community dealing with cognitive patterns [55];

8. the work of Heinze et. al. in the development of intelligent agents for military simulations [78, 74, 73, 71, 169].

Together they help to shape and guide the research by focussing research in particular directions, by dictating assumptions that constrain the thesis and by providing insights into the relevance of particular approaches. Ultimately this thesis aims to provide an approach to modelling intention recognition whilst at the same time addressing some of the gaps that exist in the state of the practice of the engineering of intelligent agent systems.

Chapter 3

Perception and Agent Design

"This is a radical hypothesis, for it implies that 'values' and 'meanings' of things in the environment can be directly perceived."—J. J. Gibson The Ecological Approach to Visual Perception

"Do I have a big neon sign above me that reads 'Slayer. Come on attack me' or something?"—Buffy the Vampire Slayer The Reborn Episode

This Chapter is composed of four sections related to the role that models of perception can play in agent system design. These sections combine with Chapter 4 to provide the basis for the intention recognition modelling framework that is described in Chapter 5.

The first, Section 3.1 explains the importance of an explicit model of perception in designing agent systems. Central to the model of perception presented is the assumption that an agent is distinguished from other types of software by the abstract (knowledge level) data it manipulates. Perception is presented as the means by which an agent converts between the data available in its environment and the more abstract representations it requires. The situated nature of agents, and importance of the environments they inhabit, is often ignored or understated by the designers of agent systems. Their focus is inevitably agent-centric, often isolating the agent from the system it will inhabit. Failing to adequately consider the environment provides little opportunity for the design process to take advantage of, or even analyse properly, the design interplay that can exist between agent and environment. Implementing perception-related agent behaviours[22] is simplified if the environment is considered during the design process and an explicit model of perception is available to mediate between the conflicting requirements of the agents to be presented with higher order abstract data and the limitations of the environment in providing them.

Section 3.2 extends a radical theory of human perception—Gibson's Theory of Direct Visual Perception [57]—to account for human intention recognition. Gibson proposes that human visual perception has evolved to allow certain invariant properties of the environment to be accessed directly. By Gibson's account, vision involves the direct perception of affordances. Or rather, that the mutually dependant nature of animals and environments

[22]Intention recognition is an example of an agent behavior that relies heavily on perception. Other such activities might include situation awareness and navigation.

has led to invariants such as *affordance* being present in the structures of the environment, and it is these structures that can be directly perceived. Gibson's theory was an early contribution to the study of situated cognition—one of the influences that moved some areas of AI research into the study of intelligent agents. Differences between Gibson's view of human perception and the accepted theories of mainstream psychology [177] are reflected in the existing approaches to modelling intention recognition and those suggested in this thesis. When combined with the model of perception presented in Section 3.1 the extensions to Gibson's theories have implications for modelling intention recognition and suggest alternative architectures and technologies.

Section 3.3 continues the focus on the importance of the environment in the design of agent systems. Five examples of environments that have been engineered to support perception related agent activities are provided. Adding labels, structures and meta-data to environments to assist agents with perception is usually the result of some ad-hoc process associated with overcoming integration problems and not some reasoned software design process. The design choices implicit in each of these examples can be elegantly expressed in terms of an explicit model of perception. This section draws out important lessons that provide yet another degree of freedom for modelling perception and intention recognition. The appropriate labelling of an environment can greatly ease the task of providing an agent with perception related behaviours. An explicit model of perception is useful in supporting the design process that determines what constitutes 'appropriate labelling'.

There is an intuitive link between perception and intention recognition. Intention recognition depends upon perceiving the results of the actions of other agents in the environment in order to recognise their intent. When introducing this thesis it was noted that intention recognition is a substitute for communication. This suggests a relationship between perception and communication, at least as it pertains to intention recognition and possibly in a broader context. Section 3.4 argues that perception subsumes communication, at least as it pertains to modelling intention recognition. Treating agent communication as just another form of perceptual input has design benefits for modelling agent behaviours like intention recognition and from a software engineering perspective there are advantages in unifying the input into an agent in a single place. These benefits are dealt with in Section 3.4 together with an critical assessment of this approach. For some agent functionalities there are benefits in treating communication as something which is perceived. The link between perception and communication is extended further in Chapter 4 where ontologies are described in the context of agent design.

The results of this Chapter are summarised in Section 3.5 and even more succinctly in Figure 3.10. When accompanied by the insights of Chapter 4 they provide the basis for six models of intention recognition presented in Chapter 5. In Section 3.5.2 the first steps toward a design methodology that might incorporate an explicit model of perception is presented. Though the development of methodology is beyond the scope of this thesis, the brief account presented helps to provide insights into the advantages an explicit model of perception offers for the design of agent systems.

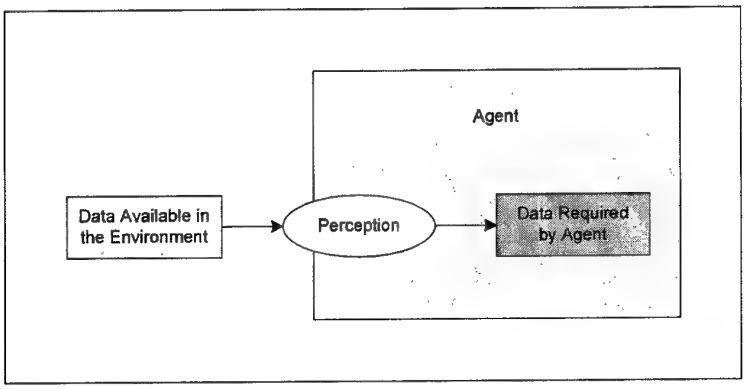

Figure 3.1: Perception is the process by which the data available in the environment is processed into the form required by the agent. What is perceived depends upon what is there and in systems where the environment is designable, it is possible to make these structures and events agent-friendly.

3.1 Exposing the Abstraction Gap

Recalling the definition given in Chapter 1, perception is the process by which an agent becomes aware of environmental events and structures [15]. Perception allows an agent to sense and make sense of its environment, a key pre-requisite to intelligent, autonomous behaviour. From a software design perspective, perception is a model of an interface between the agent and the environment. This interface includes the sensing of the environment and subsequent processing into the higher order concepts required by the agent (See Figure 3.1). Maintaining an explicit representation of the interface between the agent and the environment provides the software engineer with a means of arbitrating various requirements. An explicit model of the interface is a necessity in agent systems due to the differences that can exist between intelligent agents and other types of software. These differences exist at design time with the types of concepts, representations and processes that differentiate agents from other types of software and also at run time where the interface links dissimilar software.

Most agent models do not include a distinct perception module, preferring to incorporate the complexities of information abstraction and data interpretation into the general functioning of the agent itself, or ignoring the issue of perception entirely[23]. If perception is included within the agent model it is often *skinny*: little more than a placeholder for the sensory data[24] that enters an agent. Any processing of the data is expected to lie within the agent[25].

[23]For example the BDI model makes no mention of perception [149]. It provides a framework for implementing practical reasoning but provides no support for reasoning about or modelling perception or anything more than the most rudimentary epistemic reasoning.

[24]Adopting the terminology of psychology, sensing will be defined as a part of the perception process that acquires data from the environment but does not meaningfully alter that data: the simple detection and transmission of the lowest-order data in the environment.

[25]An obvious exception is robotics where the task of perceiving the real-world is usually delegated to a machine vision sub-system that feeds a symbolic representation of the environment to a decision making module.

Human perception is the means by which the world is sensed and conceptualised through a process of abstracting the relevant features, relationships and properties from the rich array of data that bombards the senses. The metaphor is suitable for application to agent systems where the gap between the agent and the environment is correctly characterised as an *abstraction gap.*

Most software does not share an agent's level of abstraction. In defining 'agency' in Chapter 1 it was noted that operating at the 'knowledge level' is a distinguishing property of intelligent agents. When agents are added to software environments that are not agent-oriented[26] there is an inevitable *level-of-abstraction gap* at the interface. Agents are expected to reason in high-level abstract ways (often as surrogates for human reasoning) but at the same time must inhabit environments filled with low level data that must be somehow sensed, perceived and abstracted. Perception is an appropriate model for managing the abstraction of data between non-agent environments and agents that inhabit them. Looked at in this fashion, perception is the model that manages the abstraction, translation, interpretation, and conversion of data from the environment into a form appropriate for the agent. When an agent is required to exhibit behaviors as sophisticated as intention recognition the abstraction gap is likely to be wide.

Example

A preview of the intelligent agent system that is described in Chapter 7 provides an example of the role that an explicit model of perception can play during design.

Consider the development of flight simulation software that will include an intelligent agent as a pilot to fly an aircraft around a virtual world. Suppose that the software that simulates the aircraft engine and fuel-usage is implemented in an object-oriented C++ module. At every time step (say 10 times per second) this module recalculates the amount of fuel remaining, measured in kilograms.

An agent is to be added to the flight simulator to pilot the aircraft. The agent must access some representation of the amount of fuel remaining in order that it make sensible decisions about when and where to land the aircraft. But what representation should the agent manipulate?

Human pilots typically do not use kilograms to measure fuel, they use pounds, so at the very least a units conversion is required [167]. There are many other possibilities however. Pilots only occasionally need highly accurate representations of the amount of fuel left. Commonly pilots would describe their fuel level with phrases like 'full', 'about half', 'plenty', 'enough for another half hour', or 'nearly empty'. The difficulty with concepts like 'plenty' is not only that they are inherently *fuzzy* but also that they are highly contextual. 'Plenty' is not a fixed amount of fuel but depends on what the pilot intends to do. A pilot about to land has a different notion of plenty than one about to take-off.

When looking at the fuel gauges pilot relate the amount of fuel remaining to their current activities and, almost subconsciously determine how the amount of fuel remaining relates.

The agent system designer must determine how knowledge of the fuel state arises in

[26]Most intelligent agent development involve the post-hoc addition of agents to extant non-agent systems.

the agent and how the data from the aircraft engine model is mapped into the higher order abstract knowledge required by the agent. If it was necessary to maintain a close correspondence between the agent knowledge and the knowledge captured from real pilots then it would be sensible for the agent to manipulate terms like 'plenty' and some mapping would be required. If there is a requirement that the agent operate with highly accurate numerical representations of the fuel remaining (as might be the case if the agent were more in the spirit of an 'autopilot') the mapping of data between into the agent is simplified. Whatever the requirements, there are a number of possible solutions. The type of data required by the agent, the perception model that delivers it, and the environment can all be modified to facilitate the process. Trading-off the design choices and making the appropriate decisions is facilitated by an explicit model of perception.

3.1.1 Sources of Insight

This chapter, indeed the entire thesis, seeks engineering solutions to modelling agent systems: the pragmatics of software engineering are taken as the yardsticks of the quality of the solution. Insight is sought in both the practicalities of experience with deployed systems and in the relative scientific literature. The solutions presented have both engineering utility and a level of psychological plausibility that has proven a useful, if not vital, ingredient in agent systems.

In seeking insights into the style of perception appropriate for modelling intention recognition two primary sources are considered. Ecological psychologists, notably Gibson, theorise that perception is a process by which the features of the environment are 'directly accessible' to the agent. This stands in contrast to the *sense-then-infer* theories that characterise both mainstream human psychology and mainstream agent design and yet is in harmony with the movement in AI toward "situatedness" that initially led to the widespread interest in agents. Five examples of the state-of-the-practice in interfacing agents with environments provides insights into the capacity of environments to be specifically designed to meet the needs of agents. These examples are interpreted in light of the explicit model of perception presented in Section 3.1.

These resulting approach to modelling perception is illustrated in Figure 3.10 using the notation for representing high-level cognitive designs used in the software patterns literature [55].

3.2 Gibson's Theory of Direct Visual Perception

Gibson's theory of ecological psychology (introduced in Section 2.4) highlights the important links that exist between organisms and environments and introduces the idea that there are structures in the world (Gibsonian invariants) that are directly perceivable [57]. This contrasted with the long established theory of perception as a two stage "sense-then-infer" process [177, 158] (See Fig. 3.2).

To apply the theory of direct visual perception to human intention recognition requires an extrapolation of the theory into areas of psychology it was never meant to cover. There

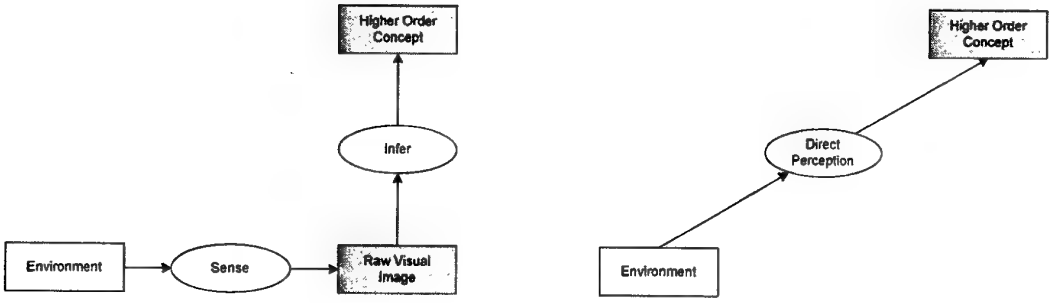

Figure 3.2: Traditional views of visual perception require a two stage process of sensing and inference. Humans cannot be aware of the true physical world except by inference based on the raw sensory data. The inference process depends on experience, memory, culture and the things that collectively define how it is that the objects that fill our world are perceived. Gibson's radical theory suggests that access is directly available to higher order structures. The strongest example of this is the idea of affordances. Gibson claims that objects can be directly perceived as a collection of the activities that they afford; hence his example of a letter box directly perceived as affording 'posting of letters'.

is some evidence in the literature to support an extension to of the theory [45, 194] but as a theory of human psychology it would be widely disputed. The following section briefly looks at the idea of explaining human intention recognition by direct perception. This is followed by an account of direct perception as a design philosophy for modelling perception and intention recognition in agent systems.

3.2.1 Direct Perception of Human Intent

The higher order structures that Gibson claims are directly perceivable refer to physical properties of objects and not to the mental states of humans. Gibson was referring to invariants such as color, shape, and spatial relationships between objects. His later idea of affordance, however, approaches something more easily relatable to human intent.

Direct perception of affordance allows a table to be perceived as a 'write at' or an 'eat here' as appropriate to the context. The affordances of an object provide the clues for future action by indicating the interactions that can occur between the observer and the object. Gibson and his colleagues were concerned with the perception of objects. But what if the idea that concepts such as affordance could be directly perceived was extended to the perception of people?

Clearly people, unlike objects, are not passive. Their affordances, if the word is even appropriate are complex, dynamic, ever-changing webs of possible interactions. In Gibsonian terms it might be argued that a stranger might be directly perceived as affording 'advice-provision' to someone who is lost, but this is an over-simplification of the complexity of human interactions.

So what would *affordances* that were directly perceived in people be like? They would give clues to the possible future actions of the person and the interaction possibilities they offered. Affordance is context dependant so just as a table is an 'eat here' if hungry and

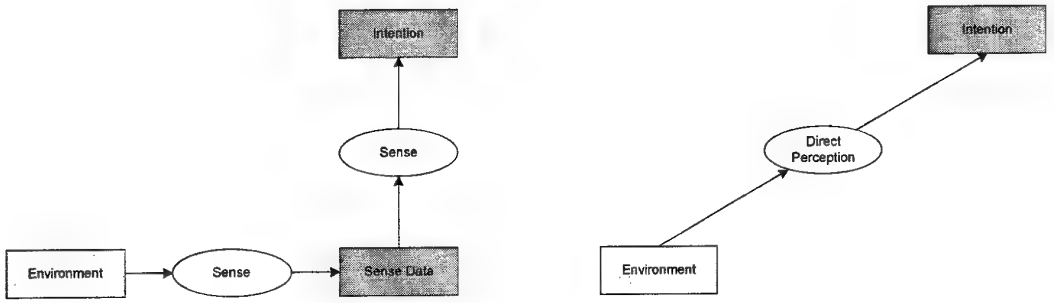

Figure 3.3: Typical implementations of intention recognition involve sensing the environment and inferring intent. Extrapolating the theory of direct visual perception leads to the idea that intent can be directly perceived. The interesting idea would be hotly debated in human psychology, but for modelling agent systems it provides insights into the possible engineering of intention recognition. From the situated cognition and ecological psychology literature comes clues about appropriate techniques, architectures and technologies for implementing this style of direct intention recognition.

a 'work at' if busy so to would people be perceived differently based on context. It is not the total set of possibilities for interaction that are perceived directly, but only those currently active ones—the ones relevant to the current context.

This description of *directly perceived human affordance* is immediately suggestive of Bratman's view of intention as partially executed, context sensitive plans that guide future action[10] combined with the style of intentional ascription proposed by Dennett [39]. Ascribed intention, specifically a view of intention similar to that proposed by Bratman, could form the basis for a psychological theory of direct intention recognition.

That Gibsonian affordance can extend from inanimate objects like tables to humans will be hotly disputed[27] but the idea offers insights into options for modelling intention recognition for intelligent agent systems.

This extension to ecological visual perception theorises that human vision is attuned to directly perceive other humans by their names, types, or features, and also by their intentions representing future action possibilities. It is not relevant to the following chapters whether or not there is any psychological plausibility to this theory. It is enough that it offers valuable insights into appropriate models of intention recognition for intelligent agents systems.

[27]Studies into human psychology suggest that the attractiveness of a prospective mate might be influenced by genetic compatibility as perceived in their physical features. Suitability for procreation is thus directly perceived in the phenotype as an invariant representation of genetic makeup. In the work of Klein, Endlsey, and proponents of naturalistic decision making [194] is evidentiary support for the view that something like direct perception can apply to intention recognition. They, like Gibson, state the importance of real environments, performing their experiments outside of the laboratory. They argue that experts do not hypothesize and test, and that recognising the situation has more to do with mapping onto memories of experiences than it does with hypothesis based inference.

3.2.2 Lessons for Modelling Agent Perception

If extensions to Gibson's theory of visual perception are used as a model for the design of agent systems then a number of approaches are immediately suggested. The pattern matching literature contains references to techniques, technologies, and algorithms that have been applied to natural language understanding, hand-writing recognition, object identification, scene recognition, and many others. These technologies, particularly when they are applied to dynamic, virtual environments offer solutions to implementing some form of direct visual perception. This casts perception in intelligent agent systems as a pattern matching task rather than a hypothesis generation, deduction, or inferential reasoning problem. In Chapter 7 a pattern matching system, Claret (see Section 2.5.4), is applied to an example agent perception task[28].

3.2.3 Lessons for Modelling Agent Intention Recognition

In the real world it is not clear that something as dynamic, uncertain and complex as an intention could qualify as a Gibsonian invariant. Experiments might be formulated to determine whether or not direct intent perception has merit in human psychology but these are well outside the scope of the thesis.

In virtual worlds, however, the idea has substantially more merit. Not only can agents actually *have* intentions[29] but because of the design control that is leveraged over the virtual environment there are known and designed relationships between intentions and data patterns in the environment. Thus the representations of intent that exist within an agent can be linked to the patterns that result from the execution of that intent either: specified by the developer at design time (see Figure 3.3); or as it executes at run-time.

The challenges are two-fold. First the development of the software must provide a set of mappings that link patterns in the environment to descriptions of intention. Secondly, there must be a run-time process to scan the environment looking for matches to those patterns. This suggests a view of perception as a *pattern matcher* that is continually fed sensory data searching for the patterns that it has been tuned to detect. Patterns might be relatively simple: the perception component of a chess playing agent playing Black might observe White opening with 'b3' and bind this simple pattern directly to the recognised intention to play the Nimzon-Larsen Attack[30]. In Chapter 7 examples of more sophisticated pattern bindings are given.

[28]There is a strong relationship between the types of processes that Gibson and other ecological psychologists believe are involved in perception (particularly visual perception) and the class of technologies of which Claret is a member (See also Section 2.5.4).

[29]Whether or not humans actually have intentions is an unanswered question for philosophy and cognitive science but certain classes of software agent make use of computational representations of intent.

[30]The reader can consult any good book discussing modern chess openings for a description of the rarely played but interesting Nimzon-Larsen Attack.

3.3 Labelled Environments and Labelled Agents

Traditional approaches to the computational modelling of perception and intention recognition have concentrated on the addition of sophisticated perceptual or inferential reasoning capabilities to the agent. The previous section showed that agent perception could be direct, and that complex higher order properties that were normally taken to be deduced, inferred, or a result of an agents reasoning could be directly perceived. The insights from direct perception support certain system architecture and design choices but do not simplify the task of implementing perception, particularly intention recognition, per se. But the environment is a designable part of the agent system. The environment can be engineered to substantially simplify agent perception. This section describes five examples of environments that have been engineered specifically to support agent perception. These examples demonstrate the capacity of the environment to be engineered and the benefits for agent perception that result.

3.3.1 Example Systems

3.3.1.1 Robot Soccer and Pitch Design

An international competition aimed at improving many aspects of robotics, Robocup soccer [98] is attracting increasing interest from the AI, robotics, and the agent research communities. Robocup soccer provides several strong examples of the impact that environmental design can play in easing agent perception. Red and blue patches applied to the bodies of the dogs afford identification of friends and foes. The ball is orange to improve visual differentiation from the green playing surface. The strongest example is the green and pink poles that mark important points around the playing arena (See Fig. 3.4). These poles serve no purpose other than to assist the dogs in the localisation task by presenting them with a static, easy to perceive, set of reference points. By adding artifacts to the environment the task of providing the agents with perception related behaviours is eased. Theoretically it would be possible for the dogs to function without these visual aids. The competition organisers have judged that benefits gained by reducing the processing load on the agent warranted the addition of the artificial aids to the environment. It is worth noting that this decision couples the agent with the environment

These poles are engineered into the agents environment to assist perception. Just like sign-posts indicating the names of streets, these poles are labels placed on an environment to supply information that would otherwise be difficult to infer.

3.3.1.2 Information Agents and Web Page Design

The world wide web emerged as a massively interconnected network of pages written using the HTML. The importance of search-engines and information agents grew as the web expanded and although the HTML supports web-page authoring and display by browsers it is not well suited to inspection by information agents, bots, or search-engines. Perceiving an HTML web environment requires an agent to parse the page and to reason about the content. Searching the web for information is invariably goal-directed in that

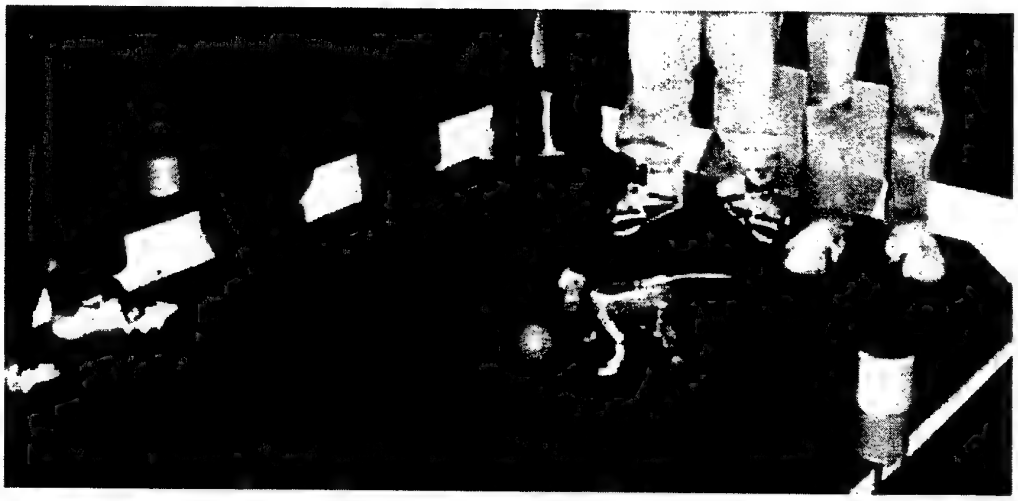

Figure 3.4: Even real world environments can be labelled to ease the task of engineering agent perception. The playing arena from the Robocup legged league. Note the presence of pink and green painted poles that assist the dogs perception subsystems localise, easing the task of orienting the dog on the playing arena.

there is query that is to be satisfied but the lack of relevant semantic content in the language structure makes HTML pages almost as difficult to search as free-text. Although there has been substantial ground gained in natural language processing techniques understanding the content of web-pages is still problematic.

One solution is to add structure to the page to assist information agents in perceiving the web. Meta-data and XML tagging allows semantic content to be added to web pages. The problem of course is that the information agent must be aware of the structures in the meta-data creating the need for standards governing the adoption of a particular meta-data set. The Dublin Core is an example of an international meta-data standard. Developed for generic resource description it provides a set of core concepts that are broadly applicable.

At present most search engines do not make use of the obvious advantages that well structured, tagged web pages can offer, because of the troublesome widespread practice of *spamming*. By adding misleading tags unscrupulous developers can trick search engines into directing traffic to their web-sites.

Figure 3.5 illustrates the design possibilities for implementing information agents. With the ultimate goal being an understanding by the information agent of the structured content of the web it is clear that the placement into the page of higher order data reduces the need for complex inference process to deduce page content.

3.3.1.3 Augmented Reality

The mobile augmented reality system (MARS) from the University of Columbia is an example that blurs the line between real and virtual worlds but provides a strong example of environmental labelling in an attempt to assist perception.

MARS is a wearable helmet system. A user walks around in the real world and the

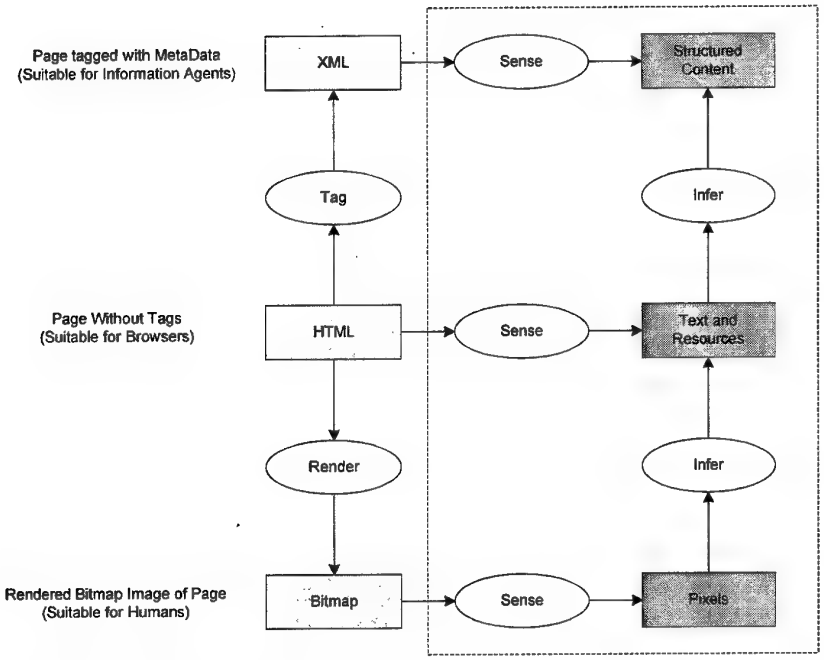

Figure 3.5: A number of options exist for implementing information agents. The web page can be rendered as a bitmap image which is then fed to a machine vision system that interprets the characters and images on the page. This extremely difficult task might seem somewhat pointless given the accessibility of the underlying HTML representation but has been proposed [101] and offers the advantage that the agent "sees" an image similar to that seen by the human. If the HTML is accessed directly then perception simpler but an inference process is required to extract semantic content from the resulting text. If the XML representation is parsed then the metadata provides structured knowledge about the content of the web page. This is a strong example of how raising the level of abstraction of the data available in the environment can assist agent perception.

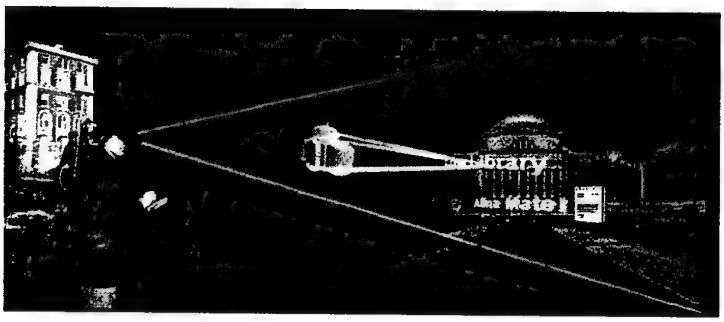

Figure 3.6: The Mobile Augmented Reality System (MARS). MARS superimposes textual descriptions onto the campus at Columbia University allowing the user to navigate and obtain useful information about the buildings and the history of the school. It is a strong example of a labelled environment, though the labelling is performed by the MARS system and the environment itself is not engineered. Extensions might see transponders attached to buildings that could provided useful contextual information.

system augments human vision by superimposing textual information (See Figure 3.6). The concept is similar to the heads-up display used in fighter aircraft that assists pilot visual perception by superimposing data from the aircraft sensors onto an transparent screen that allows a pilot to maintain an 'out-of-the cockpit' view.

MARS in isolation is a device for augmenting perception. But with this type of perception aid the environment could be engineered to assist the MARS unit. Transponders could be attached to buildings that would react to the presence of a MARS unit and transmit relevant information about the building for display to the wearer. Thus a MARS wearer might look at the train station and automatically receive an update of all the trains that are to leave in the next five minutes. The system, though human focussed has many of the properties of intelligent agent systems relevant to this thesis[31] and is yet another example of engineering environments to assist agent perception.

3.3.1.4 Why Virtual Fighter Pilots Fly into Virtual Mountains

It is often difficult to place computer generated forces into existing military simulators. This is true of air, land and sea domains and has been widely reported in the literature [140]. One of the difficulties is related to the ability of the computer generated force to perceive the environment. Observing other aircraft is usually not problematic for intelligent agents. Models of sensors (radars, eyesight and others) provide information about other aircraft in a form that computer generated forces can deal with [80]. But the environment: the clouds; the ground; the buildings; and the mountains, are second order entities in the simulation and detecting their existence and reasoning about them is often impractically difficult. This leads to the often observed phenomena of computer generate forces being blind to some objects in the environment.

[31] Note to self. I haven't seen it mentioned anywhere but imagine the cool stuff possible with interactions between two MARS units. I look at someone and their MARS unit tells my Mars unit that they are busy and don't want to be disturbed. This then gets displayed on my screen. This is computer assisted human intention recognition. How the other guys MARS unit knows that he is busy is a tricky problem but I can imagine solutions.

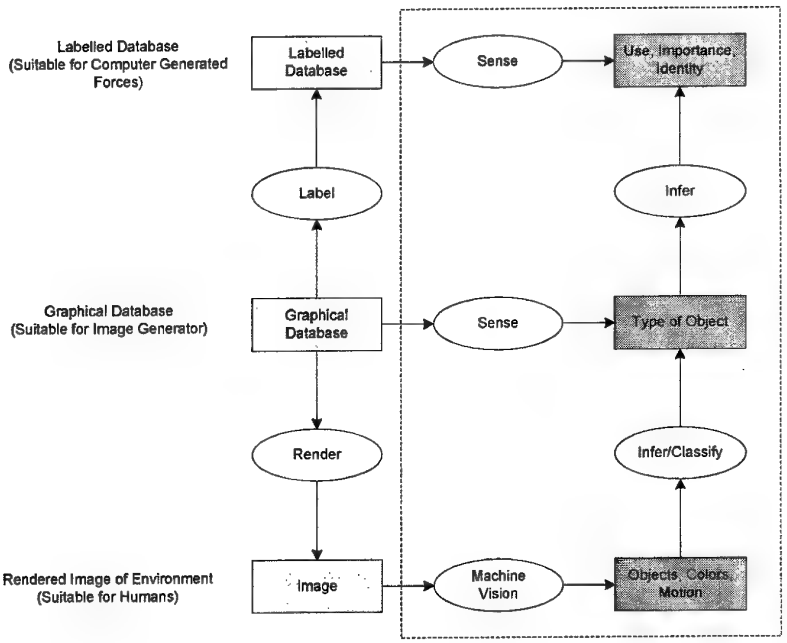

Figure 3.7: The graphical databases of flight simulators can be labelled to increase the ease with which agents are integrated into the virtual environments.

Historically this is due to these entities existing in the simulation to provide a rich and engaging visual experience for a human involved in a training exercise. The representations of these environmental elements (perhaps in some graphical database) is designed with pixel rendering in mind and shortcuts, designs, and constraints are imposed by the image generation capabilities. Converting the graphical representations of these objects into a form that is accessible by the computer generated forces is a challenge.

Solutions can involve constraining the computer generated forces to operate only in a range of conditions where the interactions with the environment are not at issue but this is not always possible. Recent initiatives by some simulation developers include the ability to "mark-up" terrain databases with information specifically for the computer generated forces. Whilst certainly a step in the right direction these systems usually offer only limited support for dynamic relabelling of the environment.

As compared with older technologies for developing computer generated forces, agents have provided considerable benefit to the simulation community. Benefits are realised in pragmatic engineering terms by reducing the development time or maintenance cost but also in providing modelling constructs and paradigms that have allowed for consideration of a variety of new issues that were previously intractable. But costs associated with retrofitting existing simulation environments with the infrastructure necessary to support intelligent agents can be prohibitive. Clearly future implementations of simulation environments and the computer generated forces that will inhabit them will need to deal with these issues. In Chapter 8 insights for the development of other systems based on the findings of this thesis are discussed and evaluated.

Figure 3.8: Computer generated adversaries in Strike Fighter are labelled to ease the game-play. This type of labelling is common in computer games.

3.3.1.5 Labelling Agents and Environments in Computer Games

Many computer games make use of labelled agents to simplify game-play. Figure 3.8 shows a screen-shot from 'Strike Fighter: Project 1'[32] displays a number of adversary aircraft that are labelled with their identity and an indication (using color and other symbology) of the side they are on and other aspects of their mission.

This idea has been extended by the developers of the game 'Black and White' to include a description of the current action of the inhabitants of a virtual world that is under the control of the player (see Figure 3.9). Not only are these labels available to the human player but they are also used internally by other computer generated characters to assist them with decision making [46].

3.3.2 Designing Labels For Agent Systems

An Agent Parable: The Saga of the Rat and the Drain-Pipe[33]

> *It was a dark and stormy night. Peter was walking down by the river. He was a gentleman of small stature but of great courage—with one exception—his rat phobia.*

[32]This game is published by Third Wire and simulates jet fighter combat between large numbers of aircraft.

[33]The following example was the source of much discussion within the Agent Lab at the University of Melbourne. The original example was provided by Dr Peter Wallis and he is honored here with the starring role.

Figure 3.9: Characters in 'Black and White' are labelled with an indication of their internal state. Not only is this information valuable for the player in determining a course of action, but the same labels are also available to other characters in the game to assist them in determining appropriate activities. The numbers are quantitative representations of the agent's internal state and the label indicates the current activity.

He feared them and hated them passionately. Rounding the corner by the boat-sheds he found himself staring directly at the second biggest, nastiest rat he had ever encountered. The rat was chewing on a piece of bread that it had found and was so engrossed that it hadn't yet seen him. Unfortunately Peter's normally stout constitution failed him, and he squealed. The rat, startled by the noise, jumped around. There they stood, 6 feet apart, both frozen to the spot, waiting for the other to move. Peter, being an intelligent fellow understood that the rat was probably as afraid of him as he was of it. But the rat's fur was standing on end and it took a couple of quick steps toward him. Peter squealed again and the rat froze. "Oh no", thought Peter, "the rat is coming to get me". Glancing around for a stick with which to defend himself Peter spied the opening of a small drain-pipe a few feet behind him. "Wheeew!", he breathed a sigh of relief. "The rat really is scared, it is just looking for a place to hide—a place like that drain-pipe—and I am standing between it and safety." Peter took a couple of big steps sideways and the rat scurried up the drain-pipe faster than a rat up a drain-pipe.

The interaction between Peter and the rat seems plausible enough. Peter (successfully) predicts the behaviour of the rat by understanding that rats are scared of humans and by recognising that when scared a rat would wish to find a place to hide. He saw that the drain-pipe represented a such a hiding place. Though Peter initially thought the rat was about to attack, the realisation that he was standing between the rat and a hiding place allowed him greater insight into the rat's intention. When he moved out of the direct path between the rat and its hiding place, the rat ran up the drain-pipe as expected.

There are lessons in this seemingly simple (though actually quite complex) interaction that illustrate some of the main points of this Chapter. To shed light on these points consider the scenario outlined above as a description of an agent system that is to be labelled to assist in the perception related tasks of those involved. Several options for labelling this agent systems exist for assisting Peter's recognition of the rat's intention:

1. The drain-pipe might be labelled **drain-pipe**. This would assist Peter in perceiving the drain-pipe but provides no extra insight into the rat's intention.

2. The drain-pipe could be labelled as **place where rats hide**. This would suggest to Peter that the drain-pipe would serve as a hiding place for the rat and provide an option to consider.

3. The drain-pipe could be labelled as **place where the rat standing in front of Peter is about to run**.

4. The rat could be labelled with **scared**. While providing insight into the rat's internal state Peter will still need to infer the rat's intention. Perhaps rats sometimes attack when scared so the intention to hide in the drain-pipe still requires uncovering.

5. The rat could be labelled as **scared and trying to hide**. Now Peter will know that the rat is trying to hide and can, upon perceiving a suitable hiding place, deduce that the rat intends to hide there.

6. The rat could be labelled with **intending to run into the drain-pipe behind you.** This is the complete and unambiguous labelling of the rat's intention. This type of label completely circumvents the need for any further intention recognition process by Peter.

The labelling schemes above each have particular qualities to be evaluated in designing the system. The first two options are static. The labels are fixed with respect to the unfolding scenario. Option 2 is less general than 1 in the sense that the labelling of the drain-pipe as a *place where rats hide* assumes the presence of rats. Hence there is a coupling between the design of the label for the drain-pipe and at least one other element in the system (the rat). Options 3 and 6 are similar in their content but applied in different place. Another possibility might be to label both the rat and the drain-pipe with a single label: the label might be applied to a relationship between two objects and not any individual label. Option 5 indicates only the general nature of the internal state of the rat without specific information about its future actions. This contrasts with option 6 which strongly indicates its future actions without commenting on the mental state that led to those action (i.e. that the rat was scared.).

A detailed discussion of the pros and cons of particular labelling choices is beyond the scope of this section. Transposing the insights from the story above into modelling intention recognition for agent systems is handled in Chapter 5.

3.3.3 Lessons for Modelling Agent Perception

Labelling environments can simplify the design of agent perception. Examples of this are plentiful yet few cast the problem as a software design exercise and solutions result from gradual evolution or pragmatic necessity with little thought given to the range of possibilities. A reasoned approach to the design of labelled environments requires an explicit model of perception (such as that described in Section 3.1) as an intermediary between the requirements of the agent and the requirements of the environment.

Dennett warns us that attaching labels to representations of the environment does not in and of itself provide knowledge.

> *"Understanding then cannot be accomplished by converting everything to the currency of mental pictures, unless the pictured objects are identified by something like attached labels, but then the writings on these labels would be bits of verbiage in need of comprehension, putting us back to the beginning again."*—Daniel Dennett, Consciousness Explained, page 57.

This problem is avoided in computational agent systems due to their *designed* nature. Dennett correctly points out that explaining consciousness as a series of labelled mental images fails due to infinite regress when these images are referenced. But agents suffer no such problem. Meaning, content, knowledge and comprehension can all be established if the 'bits of verbiage' are expressed as concepts in the agent's ontology. Or more correctly, the agent's ontology provides an explicit a-priori knowledge level theory that grounds perceptions of the agent's environment. Critics will argue that this dilutes the agent version of words like knowledge, meaning and consciousness. Answering this criticism can be attempted in two ways. First, and most controversially, by claiming that there are of course conceptual labels in the world that human perception is attuned to and mirroring this in agent systems is perfectly valid—the argument from ecological psychology and situated cognition. Second that the construction of an agent system is an activity that *requires*

the software engineer to provide grounded knowledge to the agent through careful design. If this manifests as "something like attached labels" then the meaning is encapsulated in the design choices related to the specification of those labels. A design process that recognises the distributed and situated nature of agents will result in a labelling scheme based on the subjective requirements of individual agents and not on global objectivity. Ultimately agents must act in environments. Within the software engineering constraints of flexibility, coupling and cohesion, reusability, and simplicity, labelling choices must reflect and support agent activity. As a general rule, labels should support the agents reasoning by providing grounded referents for the agent's intentional states. If this is achieved the symbol binding problem is avoided, along with the intricacies of inferential reasoning. The software engineer *designs out* these problems rendering the agent system both simpler and more abstract.

3.3.4 Lessons for Modelling Agent Intention Recognition

These examples from real and virtual worlds where perception can be assisted through suitable design of the environment provide insights for modelling intention recognition for intelligent agent systems. In simplest form an agent can be tagged with its currently executing intention and this tag can be read directly by another agent.

Not all agent have an internal representation of intention with which they might label themselves. In these cases the intention must be ascribed. The point is that this ascription need not necessarily be undertaken by the observing agent but can be performed by the software engineer at design time. This substantially reduces run-time complexity but requires supporting design-time models.

Alternatives to 'intention tagging' are available. Perhaps the 'actions' that the agent takes in the world could be tagged so that these might be directly perceived. This removes the direct access to intent but simplifies the process of inferring intent. These ideas and others and the implications of particular design choices are explored further in Chapter 5.

3.4 Perception Subsumes Communication for Modelling Intention Recognition

In introducing this thesis it was noted that intention recognition is a human activity that substitutes for communication. By observing the environment and inferring the mental states of others it is possible to gain predictive insight into their actions without requiring them to communicate. The link between perceiving the environment communication begs question about the nature of this relationship in agent systems. Taking a situated view of agency, emphasising the importance of environments, suggests a design approach where perception subsumes communication. A design stance that includes agent-to-agent communication into perceptual input has several advantages.

Perception has already been presented as the means by which environmental data is abstracted or otherwise transformed to meet the requirements of the agent. The definition of perception is now extended to encompass the translation of agent communication

into the concepts appropriate for the receiving agent. One of the challenges facing the agent community is the issue of agent communication languages (ACL). Efforts are underway to standardise agent languages to facilitate interactions between agents. If all agents speak the same language, then interactions are eased significantly. Widely adopted agent communication standards are problematic given the differences apparent in the systems that must now communicate in an environment where large networks are proliferating. Research and development into local or shared ontologies is based on the belief that universal agent languages are unrealistic and solutions lie in the development of translation mechanisms. Perception is proposed as the module that also handles the parsing, translation, and understanding of agent communication. Perceiving the environment was ostensibly about the abstraction of low-level data into appropriate forms. Understanding and assimilating the content of communication from another agent is primarily a translation activity. Strictly agents do not need to send messages formed with knowledge-level concepts but the assumption that they do is derived from the definition of agents as knowledge level entities. As a general design principle several researchers have commented on the desirability of high-level agent communication [35, 112].

Some agent systems, particularly those that require data fusion (sometimes known as multi-sensor integration) make use of communication from other agents as one of their data sources. In these cases, or in any system where agents receive information via communication that might assist, replace, augment, or inform perception of the environment it makes sense to handle that communication within the perception module. By processing the data in a single place it is possible to present the agent with a fused, coherent representation.

If all data processed by the agent are handled by a single logical module then the design is simplified. The perception module provides the complete specification of all inputs to the agent. This improves modularity and decreases coupling. For example a digital assistant's[34] perception module might provide the knowledge that "Loretta will fly to Melbourne tomorrow". For the purposes of the functioning of the agent it does not matter how the agent gained the information: perhaps it was told by Loretta; perhaps it was told by Loretta's personal digital assistant; or perhaps it inferred the knowledge by reading Loretta's e-mail and on-line credit card purchases. It doesn't matter if the knowledge was communicated by a human, by another agent, or perceived by observing the environment: in all cases the knowledge of Loretta's travel is the same.

There are some applications where it might make sense to incorporate communication into the model of perception and to integrate all data entering the agent (regardless of the origin). This treats communication as just another source of information about the world.

Applications that will benefit from a model of perception that subsumes the receipt of communication are those where interpreting state of the world is important. And that world is rich, both in terms of communication from other agents, and data from other sources.

[34]For a good example of agent based digital assistants see the Electric Elves development of Pynadath et. al [145]

3.4.1 Lessons for Modelling Agent Perception

Perception is about interpreting the environment. Communication from other agents is a valuable source of information about the outside world and agent designs that allow for the integration of information from all sources are preferred. Adding communication to the set of information sources that are processed by a model of perception allows information gained from other agents to be fused with data obtained from observing the environment. The agent can be designed as responding to a view provided by the perception module without considering what the origin of that information is. It doesn't matter at design time if the agent becomes aware of some piece of information by observing the environment, or as a result of communication from another agent.

Agents can communicate about the environment and about objects, features, or events, that might otherwise be observed. If communication is reliable and an agent is trusted then communication about an object can remove the need for it to be perceived. For the purposes of agent design it might not matter whether an agent is told about an object or if it senses it by itself. In these applications it makes sense to incorporate communication (at least as it pertains to events, features, and information that would otherwise be sensed) into the model of perception.

3.4.2 Lessons for Modelling Agent Intention Recognition

In the simplest possible implementation of intention recognition agents simply communicate their intent to each other directly. Though not really *recognition* in any strict sense it is a perfectly adequate means of *modelling* intention recognition. The result of successful intention recognition is that an agent becomes aware of the intent of an other. In human terms intention recognition can substitute for communication. In agents the relationship is bi-directional. Not only can intention recognition substitute for or assist communication but communication can assist or substitute for intention recognition. This highlights a design difference between requiring intention recognition in agents as a simulation of human behaviour and requiring intention recognition in agents as an aid to the general functioning of the agent. In agent systems where intention recognition is mimicking the human equivalent (i.e. military simulators) and direct agent-to-agent communication is available then communication might be used to model intention recognition.

Some implementations of agent intention recognition might make use of information gleaned from the environment to supplement partial, ambiguous, or unreliable communication. In this scenario communication provides some of the evidence that must be considered when inferring intent.

In other implementations agents might communicate partial information about their intent or, as was indicated in Section 3.3, the agent might be labelled directly with its intent. This creates a system that is from the perspective of the recognising agent functionally identical to the complete communication of intent.

3.5 Bridging the Abstraction Gap: An Explicit Model of Perception

Designing agent systems requires that consideration be given to the abstraction gap that separates agents and their environments. An explicit model of perception allows the agent system to be modelled and design choices made by providing an intermediary between the agent and the environment that can hold and process data as necessary. In general terms the interface design issues that can be addressed by an explicit model of perception in one of four ways (See Figure 3.10):

1. Design the agent to reason with low-level data that can be directly sensed from the environment. This effectively removes the abstraction gap by lowering the agent to the level of the environment. Unfortunately this approach removes one of the primary advantages of agents: their level of abstraction and ability to reason with abstract data. For agents that must exhibit intention recognition this is unacceptable. Or design the environment (other software components) to operate with the types and abstractions of data used by the agent. This is seldom possible in all but the simplest settings and an assumption that this is possible is one of the failings of many proposed approaches to developing agent systems.

2. Design a perception module capable of 'recognising' patterns in the environmental data and thus providing a direct mapping between the low-level data and the higher order forms.

3. Engineer higher order representations of the low-level data into the environment. These high-level representations can then be directly sensed.

4. Sense the low level data and utilise a reasoning process to infer the higher-order structures.

In application the four designs described in Figure 3.10 circumscribe a space within which solutions will be found. In Chapter 5 the lessons of this Chapter are applied directly to the task of modelling intention recognition.

This chapter has developed a view of perception as a process within the agent that transforms data from the environment into the form required by the agent. Characterising it in this fashion suggests that it is a software interface. In standard software development interfaces between system components must often convert data[35] but intelligent agents typically operate at the 'knowledge level'[125] and manipulate higher order concepts than those found in other varieties of software. Thus the interface between agent and environment must, in addition to other data transformations, abstract the data or otherwise transform it into knowledge level representations suitable for agent reasoning. Perception is a perfectly appropriate metaphor for this process in intelligent agent systems. Not only does it accurately reflect the nature of the interface but it carries the anthropomorphism often sought when describing intelligent agent concepts.

[35]Language translation, unit of measure conversion, axes translation, are all examples of routine data conversion functions required in software interfaces.

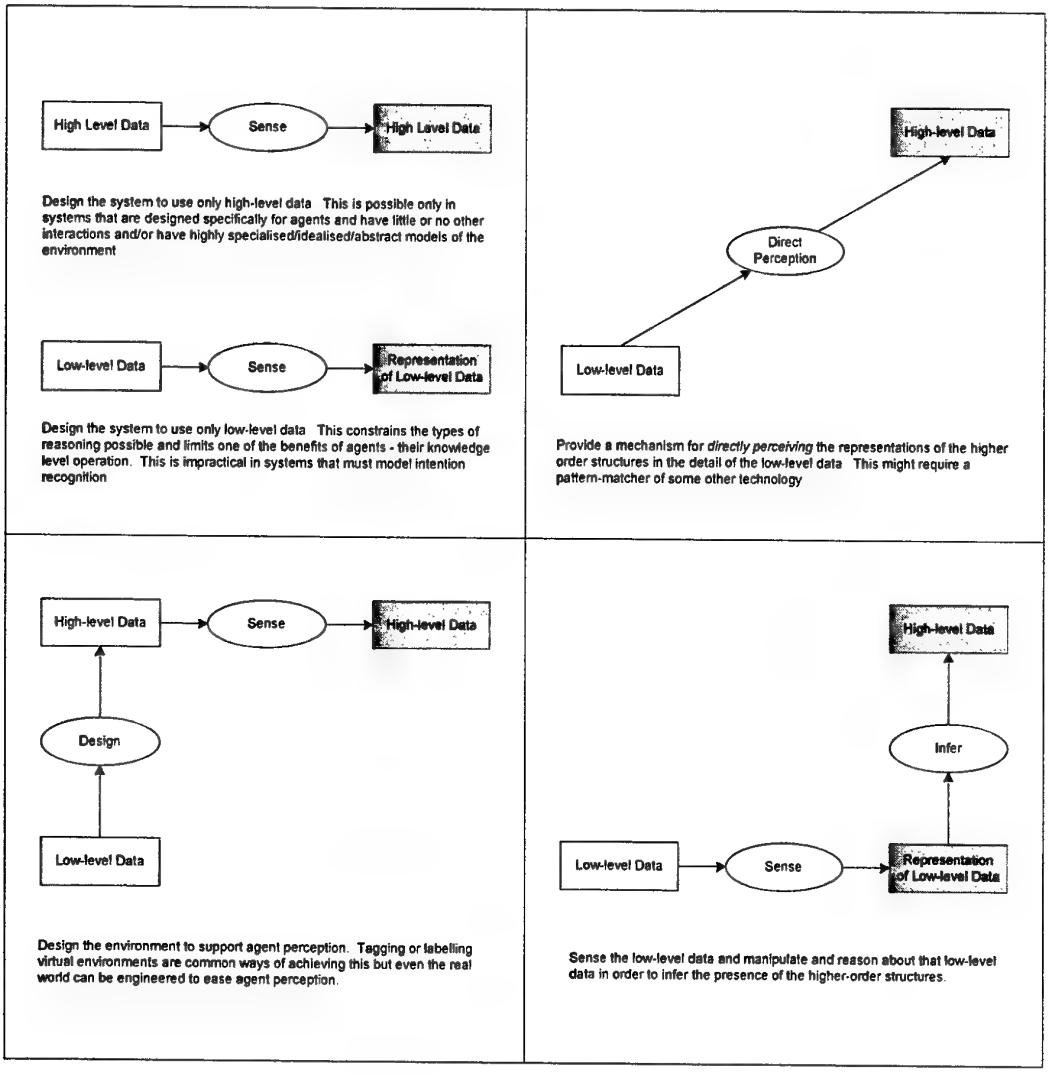

Figure 3.10: Bridging the abstraction gap. A number of plausible options for bridging the abstraction gap are available.

3.5.1 Advantages of an Explicit Model of Perception

The task of designing agents often focusses on the provision of functionality *to the agent.* This design fixation precludes the possibility of exploring the capacity of other elements of the system to designed to specifically support the required agent functionality. Intention recognition is part of a broad category of behaviours that are linked closely to capacity for the agent to perceive events and structures in the environment. An explicit model of perception provides a means for the software engineer to consider many aspects related to the provision of perception related behaviours to agents, and offers several advantages.

Perception Integrates the Environment into Agent System Design A model of perception provides a useful way of arbitrating the design interplay between the environment and the agent. As a design time aid it allows consideration of a number of design alternatives that, with respect to perception related functions, might not have been previously considered.

An explicit model of perception offers insights into design solutions and allows for the more flexible consideration of the design trade-offs available when developing agent systems. An important consideration when developing intelligent agents is designing the means by which they become aware of the events and structures present in their environment. If the environment is complex, or if the information available in the environment is in an inappropriate form, then a mismatch exists between the information required by the agent and that which is available.

In practice this information mismatch is solved in a number of ways. Conceptually it is common to think of an 'interface' or an API to an agent that allows it to connect to other systems. This interface provides the mechanism by which data enters the agent and typically there will be some manipulation or translation of data within this interface. This model of situating an agent in an environment is suitable if the information mismatch is small. If the environment is sufficiently rich or if there is a significant mismatch between the information available in the environment and that required by the agent this mapping is not straightforward.

There are other mechanisms for dealing with the mismatch however. It might be possible to design into the environment information structured in the form required by the agent. Meta-data within web pages is an example of an effort to provide constructs in an environment in a form appropriate for information agents. Just as the environment can be pushed conceptually closer to the agent so too can the agent be pushed closer to the environment. This may compromise agent design [116], or it may force the adoption of agent architectures that function directly with the data gleaned from the environment.

The mismatch can also be removed if all of the system components utilise the same set of concepts. This drives research into standard or sharable ontologies. Finally a sophisticated mapping function can convert from the form of the information available in the environment to the form required by the agent [72].

Within this landscape of agent–environment data translation possibilities lies a design space that requires careful modelling. The task is to determine the concepts

required by the agent and those that are to be made explicit in the environment. If these are different—as they almost inevitably are—then some way of resolving this mismatch is required. An explicit model of perception provides the necessary conceptualisations to assist these decisions.

Perception Allows for a Separation of Agent Modules Human perception seems to be a fundamentally different type of phenomena than human cognition. The demarcation between perception and cognition is not always clear and there is no agreement in the psychology literature about where a line (if any) should be drawn. Mirroring this uncertainty the AI research community has developed disparate technologies to simulate different aspects of intelligence [120]. Or conversely, it might be argued that the AI community has been inspired and guided by the different qualities of aspects of human intelligence to explore and develop different technologies. Whichever is correct it seems likely that the future of AI will require a compositional approach to intelligence that allows a variety of technologies to be integrated(see Section 1.3 for more details). Designing intelligent systems comprising hybrid technologies requires models that distinguish the various components of a intelligence to which the different technologies will be applied. An analysis of just what these components might be is beyond the scope of this thesis but a perception module seems like an intuitive, inevitable, and appropriate first step. At the very least it allows the segregation of perception and cognition—two of the more important concepts of human psychology.

AI and cognitive modelling aside, modularity and encapsulation are desirable properties of software and a division between an explicit model of perception and the reasoning of the agent is appropriate. Determining where this division is made is not simple but is one of the tasks of design that is facilitated by an explicit model of perception[36]. This thesis sidespteps any argument over the scientific credentials of particular theories preferring to allow flexibility in design and advocating the means by which particular theoretical approaches might be incorporated into the software engineering process.

Perception is Relevant to Simple and Complex Systems Perception might be as simple as an HTML parser and a substring matcher for an information agent, or as complex as a stereo-camera based vision system with object detection algorithms for a robot. Importantly the term perception can apply to both and has design-time value in both cases. In Section 3.3 five agent/environment pairs are described in terms of an explicit model of perception It will be shown that over a range of cases not only can an explicit model of perception describe existing implementations but it makes apparent the range of alternatives.

Perception Models a Software Interface There are good standard software engineer-

[36]Robotocists tend to draw the line at the boundary of the different technologies used to build the robot. Machine vision systems are purchasable as COTS modules that can be integrated with other AI technologies. Chalmers, French, and Hofstadter [26] and other AI researchers believe that perception and cognition are intimately intertwined and cannot be separated. Or more accurately cannot be separated at the point where AI is prone to separate them. They propose a model of high-level perception which is inextricably linked to cognition but are happy to separate cognitive activities from low-level perception. Their argument is not that a line between cognition and perception can't be drawn, but exactly where it is drawn. Indeed an adherence to the Chalmers model of high level perception could be one of the influences on the perception/cognition demarcation issue.

ing reasons for modelling perception. Perception, as defined in Chapter 1, is an interface between two software modules. Interfaces and the architectural design of systems always present headaches for the software engineer. Not only are most software bugs found at interfaces but integration issues, the bane of software project managers, are one of the primary causes for delays in software development [178]. Techniques for managing the complexity of interfaces are valuable as part of any software engineering process. Agents present particular interfacing challenges due to their autonomy, level of abstraction, and their common employment as post deployment additions to large existing software systems. An explicit model of perception facilitates the architectural design by making clear the assumptions, constraints and requirements associated with the agent/environment interface.

Perception Has Anthropomorphic Intuitiveness Anthropomorphism is one of agent technology's strengths and agents are often endowed with human qualities [25]. Many agent researchers have noted that developing agent software is assisted by adopting concepts that can be easily related to human behaviours and consequently perception is a concept considered an important part of many definitions of agency. The term perception evokes appropriate metaphorical weight and invokes an intuitive semantics entirely appropriate for describing the process by which an agent interacts with the environment.

Perception Encourages a Knowledge Level Approach An explicit model of perception encourages the software engineer to specify the knowledge level concepts that the agent should manipulate rather than simply adopting the data structures that were provided by the environment. In doing this it facilitates the development of agents that manipulate more abstract data. By encouraging the specification of agents that utilise knowledge-level data the way is cleared for the incorporation or development of ontologies. If a standard or shared ontology is to be used this will manifest as a set of constraints over the types of data that the perception module must provide. Allied to the adoption of ontologies is the idea that the reasoning processes of the agent can be raised above the lower level processing of the remainder of the system using the perception module as the means by which the agent-proper is segregated. This ensures that the agent manipulates only the high level data appropriate to it—an agent based interpretation of what some researchers have called "maintaining ontological purity" [110].

Perception Subsumes Communication Perception not only provides a way of managing the complexities of interfacing agents and environments but it is also an appropriate mechanism for translating communication between agents. Including communication as a type of perceived data simplifies tasks that require integration of different types of data and in some cases provides for more modular designs.

3.5.2 Toward a Modelling Methodology

The exploration of modelling methodologies is beyond the scope of the thesis but a brief account will make clearer the importance of the environment in conditioning the design of the agent. It will be shown in Chapter 4 that a modelling methodology like that

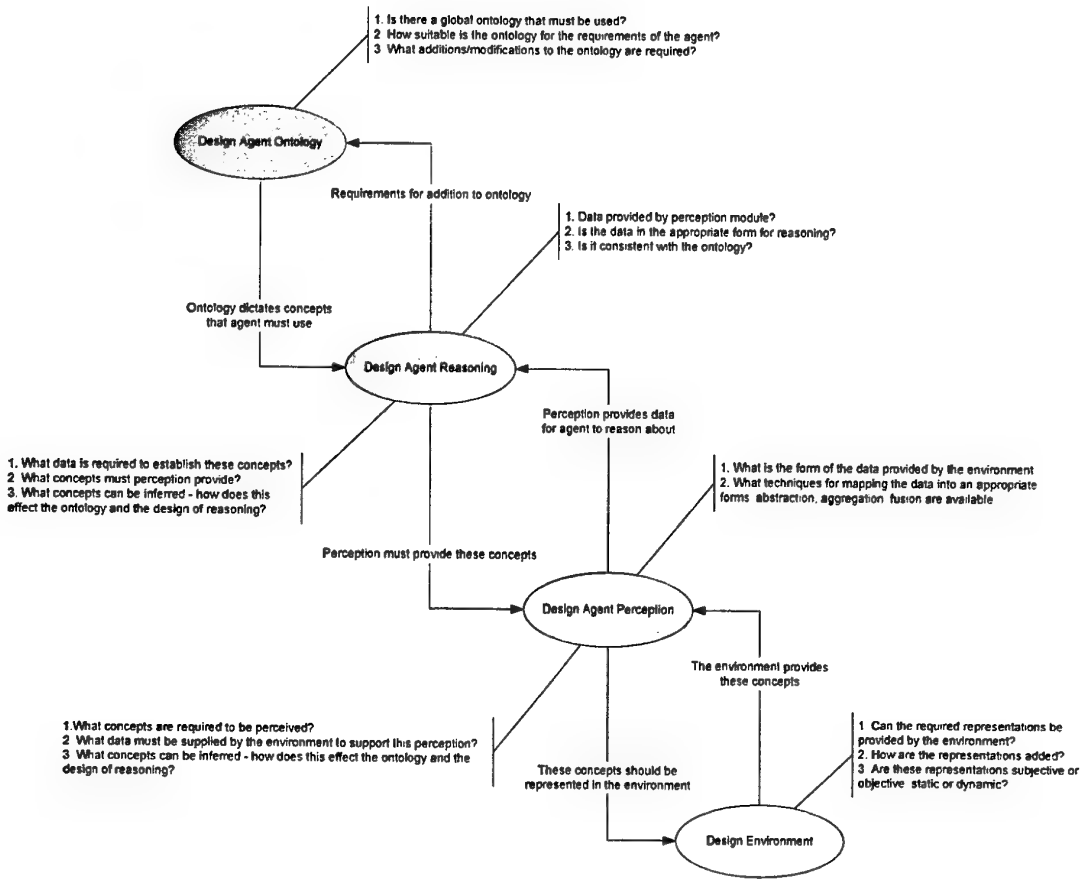

Figure 3.11: Perception can be used to assist in the design of an agent system by arbitrating the design interplay between the agent and the environment. A design process can iterate through this diagram asking and answering the questions and converging on a design solution.

shown in Figure 3.11 can incorporate the design and development of an agent's ontology into the software engineering process. This is important in dealing with knowledge level concepts (like intention) that might in some cases, be communicated between agents.

Figure 3.11 indicates an iterative process that considers the requirements of the ontologies, the agent, the perception process, and the environment. Theoretically the process may be commenced at any point, though in practice experience has shown that a consideration of agent behaviour is a suitable starting point [70].

Chapter 4

Ontologies and Agent Design

"All this talk about ontologies reduces to 'Use sensible variable names.'"—Michael Papasimeon, personal communication, 2000.

"All this conjecture—the 'ontological shock' that you speak of, for which we are so ill-equipped—is not only false but dangerous."—Fox Mulder, The X Files: 'Patient X' Episode

This chapter provides a definitive account of ontologies as the explicit specifications of the concepts that characterise an agent system and elaborates the important role that ontologies play throughout agent system design. A consequence of the model of perception introduced in Chapter 3 is that it enforces, or at least encourages, a more reasoned approach to determining the concepts to be represented in the agent: concepts including those that are perceived from the environment and those that are a result of communication from other agents. This set of concepts *is* the agent ontology, or more correctly, is a *part* of the agent ontology.

That ontologies are an important part of the specification of agent systems is by no means accepted, though Gao and Sterling [54] and Zini and Sterling [193] provide strong evidence of the utility of ontologies in agent system development. Similarly Guarino [66] cites several reasons for adopting an ontological view of knowledge based systems. Few researchers actively consider ontologies as an artifact of the agent design process and so the literature describing agent design focuses on the impact of ontologies on agent design or on ontology design as separate process.

Though not central to modelling intention recognition, the description of the means by which ontologies assist the development of agent systems is a valuable by-product of the perception-based the modelling approach adopted in Chapter 3. Agent communication is commonly expressed in a vocabulary specified by an ontology. That intention recognition can substitute for communication in social activity led to the insight that the results of the intention recognition process might also be expressed in terms of the agent ontology.

Section 4.1 defines ontologies in the context of agents and agent design and reconciles some differing views of ontology. The definition of ontology presented is an amalgamation of the various views expressed in the literature but there are subtle clarifications of detail that result from a focus on agent systems design.

Section 4.2 builds on the foundation of Section 4.1 by discussing the role that an explicit model of perception can play in acting as an ontology translation mechanism. This section extends the model of perception introduced in Chapter 3 by presenting perception as the software component responsible for ontology translation.

Section 4.3 discusses the implications for modelling intention recognition. Adopting ontologies for design requires techniques for decomposing knowledge about the intentions to be recognised. At the highest level an intention might be simply described by word or phrase, and indeed this might be all that is required for the purposes of intention recognition. Decomposed into parts and elaborated in greater detail an intention becomes the description of past and future actions, participants, goals, events, objects, patterns, data and any descriptive knowledge necessary. An ontology that describes the intentional states of an agent is a prerequisite for modelling intention recognition but has properties that have wider application and are compatible with the situated cognition literature that has flavoured much of this thesis [70].

Finally Section 4.4 shows that the consideration and development of ontologies is integrated into the agent system design process. If the ontology is predetermined—as might be the case with the adoption of a global, shared ontology—then the result is a set of constraints over the agent design. When the agent is constrained to manifest a particular ontological commitment perception plays the role of translator between the domain ontology and any preferred internal ontology. If however the ontology is not pre-specified and is one of the products of design then an explicit model of perception play an important role in developing the ontologies that characterise the agent system. This provides an approach to ontology development that is tightly coupled with standard software engineering. Designing ontologies is an interesting and growing field of computer science. Interesting, because it touches on the essence of good old fashioned AI, the structuring and representation of knowledge. Growing, because as large networks proliferate in an e-commerce environment, the computer arbitrated interchange of knowledge between diverse groups is becoming central to the business world. An engineering approach serves to demystify the "black-art" of ontology design [175] by treating it as just another software engineering activity and not as an independent discipline.

4.1 Ontologies and Agents

Many researchers have proposed definitive accounts of ontology but Gruber's has the advantage of being widely accepted, succinct, and useful:

an ontology is a formal specification of a conceptualisation [63].

Exactly what this definition means in the context of agent design will be elaborated in this section by comparing and contrasting the ontology literature with an eye to providing a clearer account of the role ontologies play in agent system design. Ontologies, as they are documented in the literature, can be classified in several ways:

Classification by Use Accepted uses for ontologies fall predominantly into two categories. The first, and most widespread, uses ontologies to provide a vocabulary

specifying the meanings of and relationships between concepts pertinent to a *domain* [66]. In this role ontologies facilitate communication by establishing a basis for a shared understanding or at least a means of translating from one understanding of the domain to another. Communication can be between humans, between software systems, or between humans and software systems. Ontologies can also specify the important concepts, relationships and structures pertinent to a *component of a system.* In this role ontologies facilitate system engineering by providing a specification of a component that aids reuse, maintenance, or requirements modelling [86].

Classification by Scope Another way of classifying ontologies is to differentiate on the basis of scope. The focus on global, shared or standard ontologies is indicative of research into ontologies intended to have wide scope. There are good software engineering motivations for preferring global ontologies. If it is possible to specify an ontology that has broad coverage then interoperability issues are alleviated. All of the software speaks the same language and communication is supported as is reuse at the component level by nature of the high degree of commonality that exists between software modules. The local, or light-weight ontology community prefers ontologies that are designed to meet the requirements of a single agent system. Local ontologies sacrifice standards and hence reuse and require translation mechanisms for interoperability but are less constrained and offer more freedom in design. There is no clear front-runner in the ongoing debate between local and global ontologies and like most software engineering evaluations it is a case of selecting the appropriate solution to meet the particular needs of the system to be built.

Classification by Existence A subtle but important and often unrecognised classification that can be made is that of existence. This is illustrated by Gruber's definition of "commitment" to an ontology as the state an agent is in when it behaves as if it had the knowledge level constructs indicated by the ontology without actually requiring the agent internally represent ontological elements at the symbol level. Rather than the word commitment it might also be insightful to consider the agent as being *ascribed* the ontology. This contrasts with the software design community that sees ontologies as *descriptions* of the symbol-level concepts that are manipulated by the agent. Some agent languages offer the capacity for a close mapping between the symbol level and the knowledge level whereas other agent implementations might be characterised as knowledge-level entities by their design only. For Gruber, and many interested in ontologies as vocabularies of discourse, agent design is often not considered. Ontologies are thought of as the set of concepts about which an agent can communicate and the internal processing of the agent is not considered. Or, if it is considered, it is secondary to the ontology that is their primary focus. Ontology design is considered as something separate and apart from agent systems design and something that deserves special consideration. Hence the consideration of an ontology internal to the agent that characterises its internal functioning and is not necessarily exposed to other system components is discounted. For the third time in this thesis it has been useful to distinguish between ascription and description (the first was agency itself and the second was intention).

Uschold [175] also classifies ontologies according to their subject matter and their formality. Obviously ontologies can present knowledge related to different topics [175] and

ontologies dealing with different subject matter are likely to be very different but this doesn't two impact on the general features of ontologies described here. The formality of an ontology is also an important issue. From a software engineering perspective the level of formality of every artifact is an open question. Mission or safety critical software requires a greater degree of formality than a script on a personal web-page. Deciding how formally specified software will be is addressed on a project by project basis so although the issue is an important one it is not central to a discussion about agent systems design but is, like the issue of subject matter, deferred until Chapter 7 when an actual system is presented.

Unifying the above classes can be achieved when it is appreciated that the literature primarily sees ontologies as the specifications of the concepts manipulated by a system in operation. But ontologies also have use for design as place-holders of knowledge that becomes increasingly detailed as the design progresses or as specifications of knowledge that act as requirements on the design of the system. As the design matures the ontology will be fleshed-out.

Throughout the design process the ontology changes and evolves, some parts of the ontology become specifications for agent communication, other parts might become specifications of the internal symbols manipulated by an agent, and still others might define the intentions to be recognised when communication fails.

Ontologies can characterise both the internal structure of an agent and a specification of the interface it presents to other agents. These ontologies need not be the same: the ontology to which the agent "commits" (to use Gruber's terminology) need not be the same as the ontology which characterises its internal operation. This distinction is useful for design because it allows for flexibility at the agent interface, it provides place-holders for the agent's internal ontology and other system ontologies and a means of converting between the two.

Ontologies should be thought of as specifications of the concepts that characterise components of a system and, in general, agent systems require at least two varieties: one that characterises the internal state of the agent; and one that characterises their external state[37]. If these ontologies are different, and except for the case where a completely global ontology is mandated, they will be, then the complexities of translation between the two must be managed. In the following section perception is presented as the ontology translation mechanism.

So in the context of agent design:

> **an ontology is a formal specification of the concepts relevant to a system or to one or more of its components.**

More specifically there are two types of ontologies that are useful in the context of agent design as it has been presented in the thesis so far.

[37]Systems have been proposed that make use of even more abstract representations. Groupings of agents into teams, communities or organisations are an example. For these systems other ontologies will be necessary. It is conceivable that a hierarchy of ontologies that matches the abstract groupings of agents may be necessary.

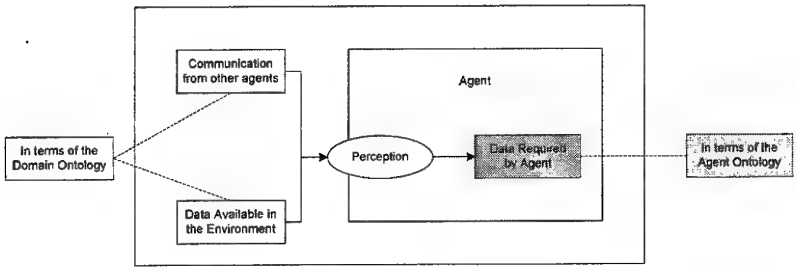

Figure 4.1: Ontologies characterise both the internals of the agent and the external interface it presents to other agents. Arbitrating between the two is one of the processes handled by perception. This view of agents and ontologies allows the design process to consider the impact of choices between local and global ontologies; unifies the view of ontologies as vocabularies for communication with the view of ontologies as systems specifications; and provides the designer with a framework for modelling agent systems that incorporates ontological considerations into the software engineering process.

The agent ontology is the specification of the knowledge that characterises the internal structure of the agent. Each agent can have a different set of ontologies if required. This knowledge is local, private, subjective, internal, and psychological. The concepts of this ontology are those that the agent manipulates[38]. Generally each agent will have its own particular ontology.

The domain ontology is the specification of the knowledge that characterises the domain. This knowledge is global, objective, external, public and social. This ontology captures the knowledge about relationships between entities and the knowledge that they share. This ontology holds the objective knowledge of the system, agents will maintain local subjective partial versions of this knowledge. This ontology is a structured aggregation of the interfaces presented by each of the system components. By *domain* here we mean a specific set of concepts that are logically grouped and a re shared by more than one agent. Within in any agent system there may be more than one domain.

These two ontologies provide enough flexibility to capture the salient features of any of the ontologies classified above. The agent ontology captures any concepts that are internal to the agent whilst the domain ontologies capture those concepts that are external to the agent. Translation between these two ontologies is the handled by perception. Figure 4.1 is a refinement of Figure 3.1 to include a reference to these ontologies.

This section has not considered the language, transport mechanism, communication protocol. It is an unproven claim that ontology development should precede a consideration of other more detailed design issues like the syntax of the language, the format of the knowledge, or particular transport protocols. Any further consideration of these issues is beyond the scope of this thesis.

[38]If this thesis were not concerned with modelling intelligent agent systems then the agent ontology might equally be described as the component ontology.

4.2 Ontologies and Perception

In the Chapter 3 the idea of perception as the mediator between the agent and the environment was introduced and used as the starting point of an exploration of the design space for modelling intention recognition. In Section 3.4 a case was made for perception subsuming communication and acting as a translator between agent communication languages. According to this model perception provides a set of knowledge-level statements the environment. Or in terms of ontologies, perception is a process resulting in a set of statements about the world in terms of the agent's ontology. If the source data is a communicated message from another agent then the process is best described as ontology translation and if the source data originates in non-agent entities in the environment then the process is closer to one of data abstraction. Additionally, as discussed in Section 3.4 a combination of these process can integrate data from multiple sources.

The definition of perception given in Chapter 1 can now be refined to include a reference to ontologies and to models developed in the previous chapters:

> **perception is the process that converts events and structures in the environment into statements in terms of the agent's ontology. Included in the events and structures are communicated messages from other agents.**

Perception takes the data presented to it (some of it in the form of the domain ontology) and transforms it into the agent ontology. The agent ontology is therefore partly specified by the products of perception. The actions an agent can take, the concepts it manipulates as it executes, and other characteristics of the agent will complete the agent ontology.

Wiederhold has proposed the use of 'mediator agents' to convert ontologies between agents [180]. These intermediary agents act as information converters, translating and converting inter-agent communication at the ontology level. It is not clear intelligent agents are the appropriate mechanism for converting ontologies. Even Wiederhold points out that the mediator agents are themselves quite simple and do not undertake intelligent interactions. A model of perception is a more natural approach to ontology translation in agent systems.

4.3 Ontologies and Intention Recognition

In the Chapter 5 intention recognition will be examined in more detail and the role that ontologies play will be elaborated in a structured way. It is useful at this point to briefly consider ontologies as they have been presented so far in the context of intention recognition.

Consider an implementation of intention recognition that requires an agent to serve queries about the nature of its intent. This casts intention recognition as a dialogue between agents that must be conducted in an appropriate language. The idea of a communicative exchange about intent implies either a shared ontology or a means of interpreting between lightweight local ontologies.

Gruber's distinction in Section 4.1 that an agent can commit to an ontology without actually maintaining any symbol-level representation of the concepts that are expressed is an important one. An agent that at run time possesses no representation of intent at all might still serve queries about its intentions. The agent will give the appearance to other agents of having intent, and hence its *intentions* can be recognised when in fact it is simply 'committing' to an appropriately designed ontology and the intentions are a result of some translation from its internal state.

Practically communication about intent is not dissimilar to an agent that visibly *labels* itself in a manner that allows other agents to recognise its 'intention' whether or not it actually intends anything. The labels that are applied to agents or other features of the environment are, or rather should be, elements of the ontology.

In cases where intention recognition is performed in what this thesis has referred to as the traditional way—by sensing state-data and by inferring increasingly abstract properties of another agent until intent can be ascribed—good design still requires an analysis of the intentions to be recognised. Analysing and describing the intentions of an agent in a way that shows the link between the state-data to be sensed and inferred higher-order properties requires a decomposition of intention that can be captured in an ontology.

However intention recognition is delivered, it is necessary prior to design to have clear descriptions of the intentions that are to be recognised. These descriptions will allow informed decisions to be made about the exact requirements of the system and how intention recognition should be modelled.

Ideally, as part of an analysis of the system performed prior to design, the intentions of each of the agents should be described and documented. It is clear, of course, that the agents might not have any explicit representation of intent in the operational system but an intentional description at design time is necessary for intention recognition to be properly considered [39]. Capturing and structuring knowledge about intention is best handled through the construction of ontologies that decompose intentional descriptions into increasingly less abstract forms. These decompositions provide the designer with the descriptions of the information that characterises ontologies at different levels and support choices about the delivery of intention recognition. It is worth making the point here that a description of an intention as it is executed is quite a different thing to the description of an intention as it is to be recognised. The two descriptions can be the same but they need not be, and in general they won't be[40]. Suchman explains this in the context of situated planning:

> *"For psychological studies, the crucial processes are essentially cognitive, located inside the head of the actor, and include the formation and effect of beliefs, desires, intentions, and the like. For social studies, the crucial processes are interactional and circumstantial, located in the relationships among actors, and between actors and their embedding situations. In either case, the problem of meaningful action is inherently subject to many ascriptions of meaning or intent, while meaning and intent*

[39] Though outside the scope of this thesis, intentional descriptions of agent systems support good design even if intention recognition is not a consideration.

[40] The system described in Section 2.3.3 allows for the same descriptions that generate intentions to be used to recognise intentions.

> *are expressible through an indefinite number of possible behaviours. Whether the final arbiter is taken to be private psychological processes, or accountability to the public world, the question to be resolved—what constitutes purposeful action and how is it understood—is the same."*—Lucy Suchman [165]

At least two types of intentional description are possible. One that characterises the intention to be enacted and one that characterise that same intention for the purposes of recognition. While only the latter is within the scope of this thesis it is important to note that the distinction exists. In practice it has been found that describing the intentions to be enacted is a natural pre-cursor to describing them as they will be recognised and so an ontology of intentional activity becomes an important step in modelling intention recognition.

The same observed actions, the same state of the world might be interpreted by agents as resulting from a different intent. And, depending on their local situation and within the confines of their context they may both be right. The fact that actions can be interpreted by different agents in different ways provides the central point of a strong argument in support of local ontologies. Furthermore, at least as far as it pertains to intention recognition, there is a case to be made for the ontological elements to reflect the purpose to which the knowledge will be put. It clearly helps intention recognition if the set of concepts about which the agent has knowledge reflect a purposive, action oriented view of the world. If the 'Saga of the Rat and the Drain-Pipe' (see Section 3.3.2) is recalled it is clear that a local, purposive labelling of the world has advantages over a global objective labelling. From the perspective of Peter's less than courageous interaction with the rat a label placed on the drain-pipe that reads 'place-where-rats-hide' better facilitates intention recognition than the more obvious, more objective, but ultimately less useful 'drainpipe'.

What is needed is a way of analysing an agent system so that agent behaviours are expressed in terms of intentions. And then a way of decomposing intent into progressively more detailed descriptions that are relatable to the intentions as they are executed, and as they are recognised. Ontologies can capturing these decompositions of intentions and structure and represent them in a manner suitable for supporting software engineering. Chapter 7 provides a brief account of one possible way of performing an intentional analysis but the detail is beyond the scope of this thesis. This idea has been developed in more detail by describing intentional analysis as an important step in the analysis and design of agent systems [70].

4.4 Incorporating Ontology Design into the Software Engineering Process

This thesis takes a counter-view to Gruber when he states that:

> *"the success of these efforts depends on the development of an engineering discipline for ontology design, akin to software engineering for conventional software."*—Gruber [63]

Rather than see ontology development as a discipline in its own right it is more natural, at least for the purposes of agent design, to incorporate it into the practice of software engineering. The need for ontologies stems from the increasingly abstract nature of the specifications associated with software. When abstraction is used to manage the complexities associated with software it creates challenges for the designer. The advent of object oriented methodologies began the move toward abstract structured software specifications. Even if it is not explicitly stated object oriented analysis is partly about ontology design [6]. Agent oriented approaches utilise more abstract concepts than object oriented analysis and design and so ontologies become even more crucial as a means of structuring, representing, and managing those abstractions.

Though not always badged as 'ontology design' software engineering is developing techniques to manage the knowledge engineering issues that pertain to software engineering. Domain modelling, an emerging pre-cursor to requirements modelling, is an important software engineering activity that establishes and structures the background knowledge necessary to construct a system. The tools of software engineering are also increasingly being used for knowledge engineering and for designing and representing ontologies [36].

Modelling agent systems requires structuring, representation, and documentation of the applicable knowledge and the processes by which that knowledge is transformed as it flows through the system. Agents are by their very nature knowledge-level entities and their designs should reflect this by maintaining knowledge-level descriptions. Agent theories, architectures and languages are to be preferred if they maintain as much 'knowledge-level' focus as is possible. If possible agents should function with, communicate with, and manipulate representations and structures that match their knowledge-level design descriptions.

Ontologies are one way of providing the structured representations of knowledge required by designers of agent systems. Ontologies provide a structuring mechanism that can index knowledge about a domain in a manner suitable for software engineering.

Simply there is no 'one true ontology' that can systematically specify a set of concepts useful for all domains and all purposes. All ontologies, however global they were intended to be, will have practical limits to their scope.

The types of ontologies described in Section 4.1 are suitable for capturing and representing the knowledge required for agent system development. These ontologies are not necessarily completely specified prior to the commencement of system development and their specification becomes one of the core activities in the design process. So ontologies are not simply knowledge specifications that are created and exist apart from the software. They are place-holders for knowledge that are elaborated throughout the design process. Ontologies specify the knowledge both inside and outside the agent and perception is the process that transforms between the two. An explicit model of perception allows a degree of freedom in the design of agent systems and the ontologies provide place-holders for the knowledge as it is elaborated during design.

For example, Figure 3.11 provided a high-level description of some of the processes undertaken as a part of agent-system design. Part of this iterative process considers the ontologies of the agent systems. It must be recognised that if an ontology is not specified as part of the agent system development then it exerts influence over the design by virtue

of the constraints that it places over the agent model. To understand, evaluate and trade-off the constraints imposed by a reused or global ontology requires a design model of an agent system that includes ontologies and makes explicit the constraints that they impose. If the ontologies are specified as part of the system development then a design model is required that indicates the dependencies and links between the ontologies and the system components in a way that facilitates design decisions. Thus the ontology design is intermingled, quite appropriately, with the software system design. If the ontology is fixed then it becomes a constraint over the design of the system. If not then the ontology will be specified by considering the requirements of the various system components and the properties of the domain.

4.5 Summary

Every software system, indeed every system, can be characterised, post-hoc, by an ontology. In some systems the ontology is more obvious than in others. In some it might be more clearly expressed in the design than it is in the executable system and in some it is barely visible at all. How seriously ontologies are teated during design depends ultimately on the importance of structuring and representing knowledge to the functioning of the system. Software that makes use of abstraction as a means of dealing with complexity is more likely to benefit from a careful consideration of knowledge structuring and hence require ontologies. If an ontology is pre-specified and mandated then it ought to be considered as a constraint over the agent-system design. If it is not pre-specified then it ought to be considered a product of the system design.

Two ontologies are useful for characterising agent systems. One that has the scope of the domain and expresses the global concepts that are useful across and between components and an agent ontology that has scope of the agent and expresses the concepts local to the agent. Perception mediates between the agent and the rest of the system by translating between these ontologies.

Ontologies can structure knowledge of intentions enabling the modelling of intentional action and the modelling of intention recognition. That these two processes are different advocates local ontologies and, if local ontologies are preferred, then a translation mechanism is required—perception fills that role.

Historical Anecdote: What is the Ontology?

When the question "What is the ontology?" is asked of an existing system the answer is more or less interesting depending upon the match between the properties of the system and the interests of the questioner. When this question was asked repeatedly by Professor Leon Sterling (a supervisor of this research) it was clear that the answers that were forthcoming where often unsatisfactory. When I personally failed to provide satisfactory answers to this question on several occasions about several different agent systems with which I was intimately acquainted it became clear that a fundamental issue had emerged. Ultimately the solution was to treat the question as one which did not demand a direct

answer but which prompted further questions: "What would it take for me to be able to answer the first question clearly and accurately?"; or "How might I design an agent system in a manner that allowed the first question to be answered?". The question led ultimately to this chapter by prompting a consideration of the role ontologies play in the development of agent system design and the role of agent system design in the development of ontologies. In this light the question "What is the ontology?" becomes highly significant. Not because it demands a specific answer but because it is one of the questions that should be considered by the designer during the development of complex software systems.

Chapter 5

Modelling Agent Intention Recognition

From *Get Smart* circa 1967

Max: "*What you're saying, Chief, is now that we know how, all we have to do is find out who, when, and why.*"
Chief: "*No, forget about where. When we find out about who, we'll know where.*"
Max: "*Well, how will how tell us where?...You see all that you've told me is that we know how, but we don't know who, when, or where. So that tells that we don't know anything.*"
Chief: "*What?*"
Max: "*Well, we know who and that doesn't tell us when, so why should how tell us where?*"
Chief: "*Max, you're driving me crazy.*"
Max: "*How?*"
Chief: "*Don't say that word.*"
Max: "*Why?*"

This chapter presents a general framework for modelling intention recognition in intelligent agent systems. It builds on, and makes use of, the insights from Chapters 3 and 4 to develop architectures that provide many possible design for implementing intention recognition.

In Chapter 3 the influence that a model of perception can play in simplifying the design of an agent system was explained and a number of options for *bridging the abstraction gap* presented (Figure 3.10). It will be seen that the models of intentional behaviour and intention recognition that are adopted result in an interpretation of intention recognition as a problem that is similarly simplified. The architectural designs that result are presented here in the style of the cognitive patterns literature and then in a more mature form as agent software design patterns in Chapter 6.

Section 5.1 clarifies the modelling of intention recognition by expressing it as a software engineering problem that will be solved by examining the system constraints in light of the assumptions and the adopted models. It is not the aim of this thesis to provide a detailed description of any particular implementation of intention recognition, but rather a broad framework for characterising and selecting from possible designs.

The models of agency, of intentional behaviour and intention recognition developed in Section 5.2 are kept as simple and as general as possible so as not to unduly restrict the possible architectures that are examined.

Section 5.2.4 briefly describes the analysis of intention in the context of modelling intention recognition and clarifies the role that ontologies play in documenting the intentional descriptions of the agent system. Some form of intentional analysis is a prerequisite to selecting an appropriate design but is outside of the scope of the thesis.

Section 5.3 develops the six design possibilities that are a direct result of combining the modelling choices of Section 5.2 with the model of perception presented in Chapter 3. Section 5.4 concludes with a discussion of the principal influences over the choice of architectural design.

5.1 Clarifying the Intention Recognition Problem

This Chapter, and indeed the thesis, is an attempt to answer the question "How should intention recognition be modelled?" from a software engineering perspective. The diversity of agent languages, technologies and applications, renders any general solution to *implementing* intention recognition futile. Broad applicability will result only from descriptions and evaluations of system designs. Consequently, the thesis focusses on modelling issues surrounding the *design* of intention recognition. Implementation details from a series of agent systems are presented in Chapter 7, but this to show the application of the architectural designs and not to explore the science surrounding a particular implementation technology or the soundness of a particular theory.

If the question "How should intention recognition be modelled?" is asked then it immediately begs a number of important questions and will receive a different response from a philosopher, a psychologist or a computer scientist. Unfortunately the answers to these sorts of modelling questions often presuppose solutions, make undeclared assumptions or are biased toward particular solutions that limit the applicability of any solution to the practical task of software engineering where scientific or philosophical soundness play second fiddle to the pragmatics of developing practical applications.

If the question is asked of a software engineer then the answer should be of the form, "It depends upon the software requirements." To the engineer the techniques of AI, the theories of psychology, or the technologies of computer science are all tools to be used in developing software. There is no universally *best* solution, rather there are a range of possible solutions with an appropriate choice governed by the particular system requirements.

This thesis provides an approach to modelling intention recognition with a focus on software engineering. Specifically the aim is to provide architectural patterns that provide options for modelling intention recognition when developing agent systems. So as not to commit to a particular technology, architecture, agent platform, language, or theory, the models are kept as general and widely applicable as possible. Presented here is a framework for modelling a range of possible designs for implementing intention recognition. This

allows the design space to be characterised in a way that provides solutions appropriate to the requirements of the system.

Informing the intention recognition design process requires the software developer to describe the intentions that are to be recognised. These descriptions (together with other pertinent system requirements) can then be analysed to guide decisions about the design of intention recognition.

Providing a framework for modelling intention recognition in agent systems requires that the problem be presented in a manner general enough to cover a useful range of application domains and allows the consideration of a range of possible solutions. To this end high-level models of agency, intention, and intention recognition are constructed. These are then combined with the results of Chapters 3 and 4 to produce a set of six architectures.

5.1.1 Assumptions

This thesis is predicated on a number of important assumptions that are presented here. Some of these assumptions limit the applicability of the designs or act as important guides to future utilisation. Some are a reiteration of those that were presented when the thesis was introduced and some are a result of the findings of subsequent chapters. They are all presented here for completeness.

5.1.1.1 Software Engineering is Important

Software engineering is an often neglected aspect of agent development. This thesis is concerned not with the development of an agent theory, an agent language, or with technology that will be used by agents. Rather it deals with the development of agents for practical applications. It is assumed that these agents will be developed according to a process and that the usual software engineering activities will be present. Agent researchers often undervalue the role that software engineering can play in informing the agent development. When research is transitioned into industry it will inevitably forced to integrate with software engineering practices. If those practices have the capacity to simplify and inform the research then they should be leveraged early.

5.1.1.2 Intention recognition is (in part) an agent architecture problem

Intention recognition can be implemented, or at least facilitated, by selecting an appropriate agent system architecture. Most implementations of intention recognition fail to consider the aspects of the system that lie outside of the agent. If the insights from Chapter 3 where unconvincing then further examples are presented in this chapter. Selecting an appropriate architectural design can ameliorate implementation problems. The selection of an appropriate architecture becomes a standard software engineering activity when the options are presented in the form of patterns.

5.1.1.3 Agent System Design Drives the Selection of the Architecture

There is an assumption that the design process is free to trade-off requirements and that a particular solution has not been mandated. Non-functional requirements such as performance, fidelity, simplicity, maintainability can influence the particular design chosen. A particular approach to intention recognition might be rejected on the grounds that it fails to meet performance requirements even though it is elegant, cognitively plausible, sound and complete, and well specified. The design process considers intention recognition as just another functionality to be added to the agent system and not an end in and of itself. The task is to build an agent capable of intention recognition and not to build the technology required to implement intention recognition. This distinction more than any other marks this thesis as different from most other agent research—it is not about developing theories, architectures, or languages: it is about developing agents.

5.1.1.4 Patterns are a Valuable Resource for Agent Developers

Published solutions to the problem of intention recognition focus on a domain, a language, or an implementation. As such it is often difficult to translate the successes into other domains. Reusing, or at least learning from, the experiences of others requires more general higher level descriptions of agent systems that transcend issues such as language dependence. To this end agent researchers should be encouraged to publish the aspects of their systems that are likely to be widely reusable. These are likely to take the form of *patterns* in the style of the object oriented software community. Unless there is standardisation among agent languages the patterns that will have the greatest scope for reuse will be those that focus on the architecture of the agent system (that is the relationships among the agents and between the agents and the environment).

5.1.1.5 Agents Should be Knowledge Level Entities

Agents should manipulate more abstract concepts that other types of software[41]. True in general, but true strongly for *intelligent agents*. That is not to say that agents won't occasionally make use of low level data but the tendency should be for the agent designer to prefer more abstract representations. In Chapter 4 it was shown that an agent ontology can be developed as part of the software engineering process and that an ontology plays a key role in defining the agent. Perception was presented as the means by which the agent translates any available data into the form defined by its ontology.

5.1.1.6 Agent Design Requires an Explicit Model of Perception

In order to maintain the preference that an agent manipulate higher-order concepts it is necessary to convert from the data that is available into the form preferred for the agent. An explicit model of perception provides the appropriate means of arbitrating between the

[41]Recall Nwana's claim from Section 2.1 that an agent should manipulate knowledge level concepts. It was an assumption of this thesis that an agent is a knowledge level entity that should manipulate more abstract concepts than other types of software.

design of data representations internal to the agent and those that are externally found in the balance of the system.

5.1.1.7 Designable Environments and Designable Agents

As discussed in Section 3.3 a surprising number of agent environments are at least partially designable. Furthermore some of the architectures presented assume that other agents can be directly modified or extended to support intention recognition. This assumption is not commonly made by researchers investigating agent behaviours. Or, more correctly, it is not made explicit. Systems tend to emerge with important design decisions hidden by particular theories or architectures.

5.1.1.8 Explicit Internal Representation of Intent

The agent who's intent is being recognised need not have any explicit internal representation of intent. Some agent languages model intention but most do not. The BDI languages are a good examples of a language with a strong explicit model of intent within the language itself and some have argued that SOAR has some form of implied representation of intent or at least provides the constructs necessary to model intention. An internal representation of intent can simplify intention recognition but the requirement that the agent who's intent is to be recognised actually maintain some representation of intent would be unrealistically strong in a thesis seeking generally applicable solutions. From a software engineering perspective there is no problem to ascribing intent to a software entity that clearly does not *intend*. Concerns about psychological or philosophical are addressed by Dennett [39].

5.1.1.9 Single Agent

Though intention recognition is inherently multi-agent this thesis will not deal with intention recognition issues associated with more complex social systems. Sonenberg and Tidhar [162] provide insights into some of these issues. Issues of ambiguity of identity, and of recognising joint intentions are left for future work. In Chapter 8 this issue is briefly revisited.

5.1.1.10 Intentions Well Understood

In order to make appropriate design time decision about the nature of intention recognition it is important that the intentions that are to be recognised are well understood and specifiable. Some process for documenting and analysing intentions is a necessary part of making appropriate design decisions.

5.1.1.11 Intention Recognition Part of Larger Functionality

The agent is expected to exhibit a wide range of behaviours and intention recognition is just one of those. Research can often focus too closely on a particular type of behaviour

resulting in agents that are *single minded.* Intention recognition must fit within a larger context and so any supporting methodology or technology must be seen as part of a larger whole.

5.1.1.12 Heterogeneous Agents

Many agent systems are heterogeneous. Agents are not necessarily architecturally similar. They do not have access to the internal mental attitudes of the other. Nor do they classify the intentions that they recognise in the same manner as the ones that they execute. When one agent intends 'to seek information about cats' another agent might recognise that intention as 'to access web pages'. Both descriptions are correct—one is for guiding action, one is for recognition. The description that an agent might place give to its own intentions might be unsuitable for another agent. Simply, they do not share an ontology. The intention recognition process must deal with this if required. This specific problem in intention recognition is synonymous with the more common problem of communication between agents that do not share the same ontology.

5.2 A Modelling Approach

To develop a framework for modelling intention recognition in intelligent agent systems it is necessary to first provide characteristic models of agency and intentional behaviour. The models are not detailed, they are deliberately high-level and general. They are not meant as formal definitions of agency, or of intentionality, but high-level models that assist in the development of design at the architecture level. As aids to architectural design the internal detail is less important than the structural form.

5.2.1 The Agent Model

In Chapter 1 the following definition was given:

> **an agent is an autonomous entity situated in an environment that it can perceive, and in which it can act.**

This description is applicable to many existing agent systems, perhaps even most agent systems. When the further requirement that the agent is *intelligent* is stipulated an additional component, here called *reasoning* is added to the agent. Associated with the reasoning module is the agent ontology[42].

The detail of this definition has been dealt with previously (Section 2.1) but there are a couple of salient points that require highlighting.

As yet the agent model includes no representation of intention (this will be introduced in the following section). In general there is no requirement for an intelligent agent to

[42]The agent ontology is a documented part of the agent model providing constraints over the design or a place holder for the iteratively developed design of the ontology (See Section 4.4)

utilise intentional reasoning. An agent that implements a model that includes some form of explicit run-time representation of intention might make some of the implementation choices simpler. Intention is seen as a characteristic of a particular implementation and not a fundamental property of agency. The real value of this model of agency lies in the separation of perception from reasoning and the association of reasoning with the agent ontology. This emphasises the importance of maintaining the knowledge-level status of the agent in the face of a requirement to interface with a system that uses far less abstract concepts. This is particularly relevant to intention recognition. Demarcation decisions between reasoning and perception are critical to good agent design and are an important influence over the models presented. The credibility of the models rely on the acceptance that an explicit model of perception as presented in Chapter 3 is a valuable concept for agent development. The need for the explicit model of perception depends in turn on the assumption that an agent should be presented with more abstract representations of data than those commonly found in its environments. This is not to preclude the ability of an agent to reason with low-level data but simply to highlight the importance of making appropriate design decisions and to provide a means of making them.

5.2.2 A High-Level Model of Intentional Behaviour

The modelling approach adopted is to describe intentional behaviour in three increasingly detailed, less abstract levels (See Figure 5.1):

The intentional level describes the intentions of the agent. It characterises these in terms of desires, beliefs, goals, plans, and other high-level intentional states. These intentions give rise, in a directly causal way, to activities.

The activity level describes the activities and the actions undertaken by the agent. These actions are a direct result of the intentional state of the agent. They might be considered as the set of actions that an agent can take. The tactics that might be adopted, the plans that might be selected, the effectors that can be operated, or the functions that might be called. These activities can be considered a decomposition of intention as well as a more detailed description of it.

The state level describes the agent by reference to its externally accessible features or features of the wider environment that are indicators of the agent's state. This is the description of the agent as it appears to other agents in its environment. Whereas intention is an internal mental state of the agent, this level describes the normally visible external appearance of the agent.

These three levels can be related to the three levels of Dennett's Intentional Stance [39] and to the three levels of human processes described by Leontiev [107]. In any case the choice to represent intentional behaviour in this way is somewhat arbitrary and there is little to impede the adoption of a different structure within the methodology proposed here should an alternate model prove advantageous.

At this stage the model makes no comment about the processes that cause intention to manifest certain activities or the process by which activities influence the state of the

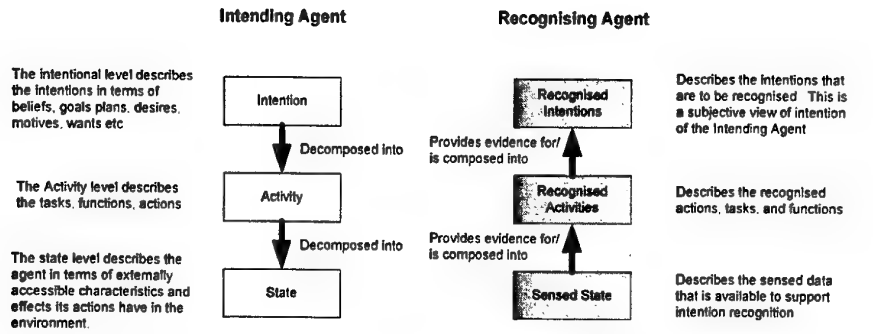

Figure 5.1: The intentional behaviour of the Intending Agent is modelled as a tri-level decompositional description capturing the salient characteristics. Intention recognition is a related, subjective (from the Recognising Agent's perspective), reversal of this process that captures the features necessary for supporting intention recognition.

environment. This is in the detailed design of the particular agent. In this respect the model is little more than a description template. Much of the literature that describes the semantics of intention in agent systems is directly concerned with the detail of the manner in which intentions result in actions [148, 166, 153, 10]. Any of these models might be superimposed onto this description template to provide a formal specification of the process if it is required. It is a feature of this model that it remains agnostic with respect to the execution semantics of any adopted model of intention.

A model of intentional behaviour is a necessary precursor to a model of intention recognition[43]. The agent literature contains many different views of intentions. Producing yet another model of intention as a competitor to these is unproductive. Each of these models, and others, has value in certain contexts. A software engineering model that spans the breadth of these models and allows the developer to describe and model systems before committing to a particular model of intention (or intention recognition in the context of this thesis) has more value. Simplicity and coverage are the hallmarks of the model of intention presented.

Modelling agent behaviour in this way serves two purposes: first, and most importantly for this part of the thesis, it supports a model of intention recognition that will eventually lead to the development of a set of design patterns; second, it can be used in the future as a description template for documenting and analysing agent systems that are to be fitted with one of the design patterns (See Chapter 6). Thus it is a model that gives insight into intention recognition in order to construct a suite of architectural designs and it is an analysis model that can be completed to inform the design of a particular agent system.

[43]It might be argued that this is not strictly true. A theory might be posited that suggests that intention recognition does not strictly require a model of intention. Though it is not clear how this might work, a subsumption architecture such as that proposed by Brooks, might be argued as having the capacity to model intention without any representation of intent.

5.2.3 A High-Level Model of Intention Recognition

A direct consequence of the model of intentional behaviour adopted is the model of intention recognition shown beside it in Figure 5.1. Intention recognition is modelled by taking a subjective view of the features that characterise the intentional behaviour. For each intention to be recognised it is necessary to describe the features that support recognition. These features occur at the same three levels but their purpose is now to support recognition and not to describe execution. The two parallel descriptions are obviously closely related but there is no requirement that they share concepts. The features that characterise a an intention as it is executed might have little commonality with those that support recognition.

Intention recognition requires the recognising agent to acquire or build a representation of the intent of the other agent. The data that supports the building of this representation is intuitively (and commonly) acquired from the sensing of the changes in the environment. Based on the model of intention presented in the previous section there are three types of data that are potentially available to support intention recognition (See Figure 5.1):

Sensed state documents the states that the agent can observe directly in the environment and the changes that are indicators of the activity of other agents.

Recognised activities describes the activities that are performed by an agent as they are recognised by another. Descriptions of the activities as they are recognised will include references to the states of the environment that are indicators of those activities.

Recognised intentions describes the intentions as they are recognised. Typically these will be described as compositions of the activity level descriptions, as sequences of recognised actions or observed events.

Intention recognition commences with the intention of the intending agent and concludes with the recognised intention in the recognising agent. Through a path, as yet to be defined information about intent flows from the top left hand box of Figure 5.1 to the top right hand box. The end goal of intention recognition is to arrive at some description of the intent of another agent by processing available information. As the recognising agent undertakes recognition it might gather and assimilate evidence at any of the three levels.

Six possible paths exist from the description of intention in the intending agent to the to the recognition of that intention in the recognising agent. At the very highest level the possibilities that exist for modelling intention recognition are therefore the six options shown in Fig. 5.2.

So the model of intention recognition takes the model of intention presented in Section 5.2.2 duplicates it, albeit in a subjective form, in the Recognising Agent and provides a number of possible pathways for the description of intent to be mapped to the recognition of intent.

If intention recognition is considered as an *abstraction gap* problem similar to that addressed in Chapter 3 then the possibilities available for traversing the sixth pathways are those exactly those that were suggested by the alternatives that were presented for

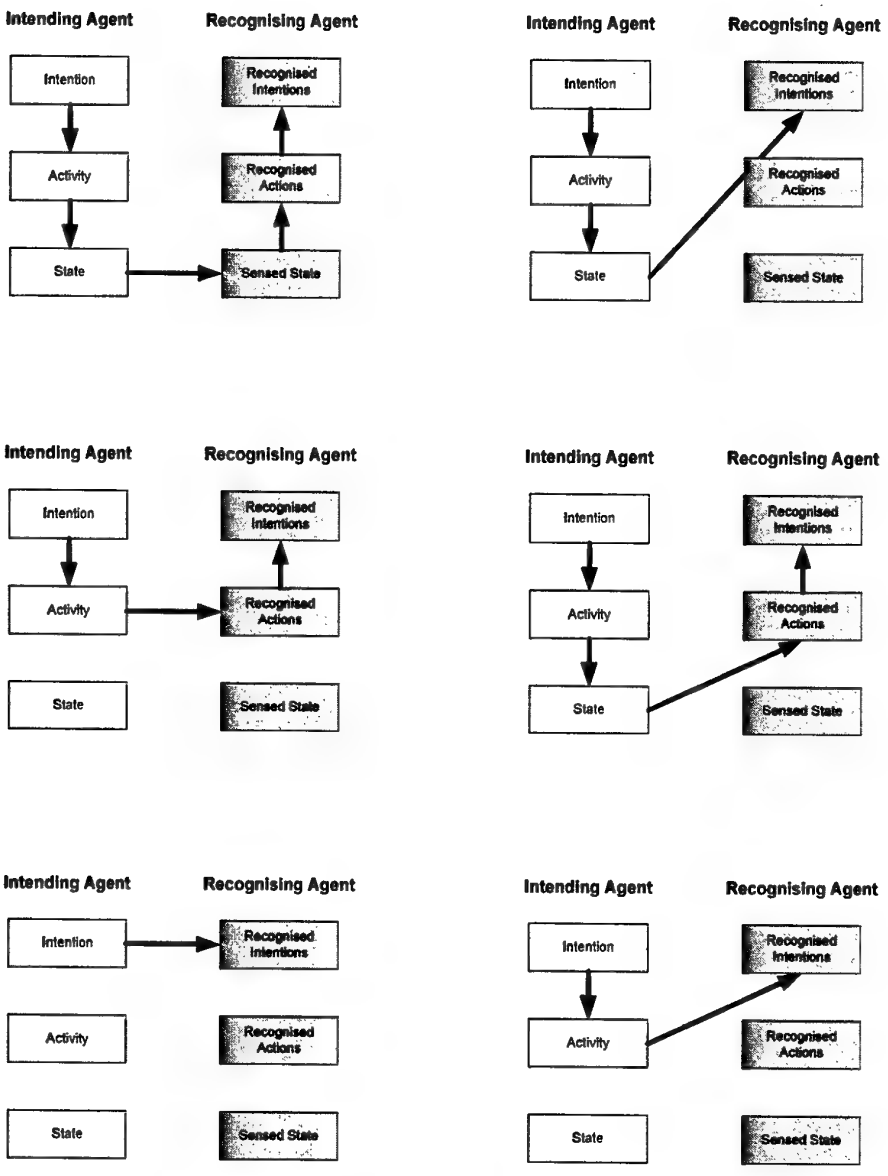

Figure 5.2: Six possible paths exist that commence with the intention in the intending agent and conclude with the recognition of the intention in the recognising agent. These pathways are the basis for the development of the design architectures presented in Section 5.3 and then in a more comprehensive format in Chapter 6.

modelling perception. Applying the model of perception of developed in Chapter 3 leads to the designs for intention recognition that are presented in the following section. Modelling intention recognition in this manner allows for the consideration of many alternatives and the detail of the designs that follow in Section 5.3 is partially provided with reference to the techniques proposed in Chapter 3 for simplifying the provision of perception.

5.2.4 Intentional Analysis

This section makes two claims. The first is that for modelling intention recognition it is necessary, prior to any design decision, to document and analyse the intentions manifested by agents in the system. This analysis should include both an objective examination of the intentions that describe the behaviour of an agent and then a series of subjective interpretations of that intention as it is recognised by others. This claim is so intuitively obvious that it needs little justification. If intention recognition is to be modelled intentions must be understood documented, and presented in a way that supports design.

The second claim is that the first part of this process, documenting the behaviour of the system in an intentional way, is useful precursor to the development of any agent system, regardless of whether or not intention recognition is a functional requirement. This claim is less intuitively obvious, but if justified means that intention recognition can leverage off the intentional analysis that would be undertaken as a normal part of the analysis of any intelligent agent system. This claim is left as an unsupported hypothesis for future research though supporting evidence is available in the literature [70].

The purpose of an intentional analysis within the scope of this thesis is twofold: to provide intentional descriptions of the behaviour of agents; and to provide a description of the intention recognition process. This corresponds to documenting the six boxes shown in Figure 5.1. Describing the behaviour of an agent system in an intentional manner (filling out the left hand boxes) is quite a different task to describing the recognition process (filling out the right hand boxes).

In the simplest possible terms, and conforming to the models of intentional behaviour presented earlier, an intentional analysis might be considered as the activities related to answering the following sets of questions:

- What are the intentions to be executed?
- What activities result from these intentions?
- What states of the environment are a consequence of these actions??

- What are the intentions to be recognised?
- What activities will indicate that the agent has theses intentions?
- What states indicate that the agent is engaged in these activities?

or adopting a bottom-up approach.

- What states can I perceive?
- What activities are indicated by these states?
- What intentions are indicated by these activities?

It should be noted that the model of intentional behaviour presented in Section 5.2.2 was biased heavily toward supporting intention recognition and may not be appropriate for documenting intentional behaviour in agents for other purposes.

Any detailed consideration of intentional analysis is outside the scope of this thesis[44] but a brief account will give pointers toward future research. The process of specifying, describing, and documenting the intentions to be recognised is a part of intention oriented analysis[70]. Techniques for eliciting the intentional descriptions from human subjects are well documented [181]. Others, such as Dennett [39], propose less structured and well defined approaches for what might loosely be described as an intentional analysis of non-human systems. There are many methods for analysing the intentions in a system. A discussion is beyond the scope of this section but Cognitive Task Analysis [181] and Cognitive Work Analysis [152] are examples. As a minimum, the analyst can resort to folk-psychological intuitions and adopt the intentional stance. More recent trends in software engineering are leading toward standard tools and techniques that are suitable for this type of analysis [155]. Use cases are a technique from Object Oriented Software Engineering (OOSE) [82] that have been adopted by the mainstream object oriented community and are now supported by the UML and many case tools. Use cases were introduced to enable the specification of requirements in systems where the interactions between the user and the system were complex. That a use case analysis commences by considering scenarios sets the it apart from other OO techniques. Scenarios allow the future user of the system to visualise the expected interactions and to describe the system in terms of the activities that it must support and the functionality that it must provide. Heinze and Papasimeon have shown that a Use Case analysis can be successfully adapted for specifying requirements in agent systems [73, 59]. There is a strong link between use cases in object oriented analysis and design and descriptions of intentional behaviour in agent systems. The diagramming techniques of the UML are appropriate for documenting the intentions of actors and agents in a software system. In Chapter 7 an example of the use of the UML in capturing intentional descriptions for the purposes of modelling intention recognition is provided.

5.3 Designs for Intention Recognition in Agent Systems

In Figure 5.2 six pathways from intention to recognition were presented. These pathways were revealed by considering the possible options resulting from the three level decompositional model of intentional behaviour and a corresponding model of recognition. The options for moving from a description of an intention in one agent to recognition in

[44] One of the assumptions in Section 5.1.1 was that the intentions of the system were well understood.

another become software architectures when considered in the light of the four alternative ways of bridging the abstraction gap summarised in Chapter 3 (see Fig. 3.10). The lessons of Chapter 3 provide the first clues about the nature of the processes that transform data between the agents and within the recognising agent. The thesis stays mute with respect to the processes within the intending agent. Not only are those processes heavily implementation dependant and well described in the literature they are irrelevant for the purposes of intention recognition[45].

All intention recognition commences with the intent of the *Intending Agent* and concludes with the recognition of that intent in the *Recognising Agent*. The process by which intention recognition occurs depends upon the particular design chosen. For each of the possible architectural designs there are several variants that depend on the particular implementation details. The basic architectures are briefly described below (See also Figure 5.3) and then in a more detailed manner in the following chapter.

The designs are presented using the notation and style of the cognitive patterns literature of Gardner et. al. [55]. This is done for four reasons: at this stage they are more like cognitive patterns than software patterns; they can be related to theories of psychology; they are better explained without the necessary detail required for software engineering; agent software design patterns are provided in Chapter 6.

Design Option 1 This corresponds to the case that is most commonly referenced in the literature, and has been described elsewhere in this thesis as 'sense-and-infer'. The intending agent executes its intentions, its actions, and ultimately influences the state of the environment. This state is sensed by the Recognising Agent and the higher order properties of actions and then intention are successively inferred. As mentioned earlier there is a series of variants available that depend upon the specifics of the implementation. For example, although the figure shows the Intending Agent as actually executing intentions there is no requirement for this to be the case. In fact the only interaction between the Intending Agent and the Recognising Agent is via the state as it is reflected in the environment and so the particular internal detail of the Intending Agent is irrelevant to this particular design. As for all of the design options here there is more detail presented in the following chapter.

Design Option 2 Design option two short-circuits the recognition process by providing the Recognising Agent with direct access to the descriptions of the Intending Agent's actions. This gives the Recognising Agent a description of the activities of the agent with which to infer intention. This architecture required that the Intending Agent be considered as part of the design process as its internal state must be made available and, if it is in an inappropriate form, it may need to be augmented or labelled with suitable representations.

Design Option 3 Design option three appears to be the simplest of all. Rather than conduct any sophisticated reasoning the Recognising Agent is given direct access to the intentions if the Intending Agent. If the Intending Agent manifests no explicit internal representation of intent then one must be provided. This is referred to as

[45]More correctly, they are irrelevant for intention recognition given the assumptions that underpin this thesis. It is not inconceivable that the dynamics of executing intentional behaviour and not just a reference to the agent state could influence the design of intention recognition.

'labelling', i.e. the agent is labelled with a representation of its intent specifically for the purposes of supporting intention recognition. If the Intending Agent actually has intention then the recognising agent might be given direct access to some version of this. Perhaps the access is given via communication, that it the recognising agent asks for a description of the intent. Perhaps the intention label is multi-valued and each agent that views the label gets its own specific information. In this fashion, although it is a simplification of the intention recognition process, the act of labelling the intending agent with a description of intent can be quite sophisticated.

Design Option 4 Design option four is an implementation of 'direct perception' or ecological perception (See Section 2.4). By conducting intention recognition completely within the perception module the agent reasoning is supplied directly with recognition of intent. This supports the preservation of a highly abstract agent ontology.

Design Option 5 Design option five is a compromise that utilises direct perception for the first stage of the intention recognition process and inferencing for the second.

Design Option 6 Design option six is similar to option 2 but makes use of direct perception to convert the sensed actions into intentions.

5.4 Constraints on the Design of Intention Recognition

In the following chapter these designs are presented as software patterns and the application and implementation of these software architectures is described in detail. Prior to the elaboration of the designs patterns it is necessary to consider some of the design constraints that will influence the applicability of these designs.

Legacy code Many agent systems of any scale are built on top of legacy code. From the introduction of mobile information agents to the web, or the addition of assistant agents to existing systems the environments in which agents operate are often in existence. The post-hoc modification of these environments is often too costly to justify. This leads to compromises in the agent design necessary to deal with a less then ideal environment.

Design Influence on the Environment Often the environments in which agents must operate a designed and developed separately and apart from the agent development. This might be due, as in the case of the internet, to the agent being developed to operate in someone else's environment. Or it might be due, as is the case in some military simulations, that the teams that develop the computer generated forces are independent of the teams that develop the rest of the system components. In cases where the agent development cannot influence the design of the rest of the system the design choices available for implementing intention recognition are diminished.

Explicit Internal Representation If the Intending Agent makes use of explicit representations of intention or action then these might be utilised to assist the modelling

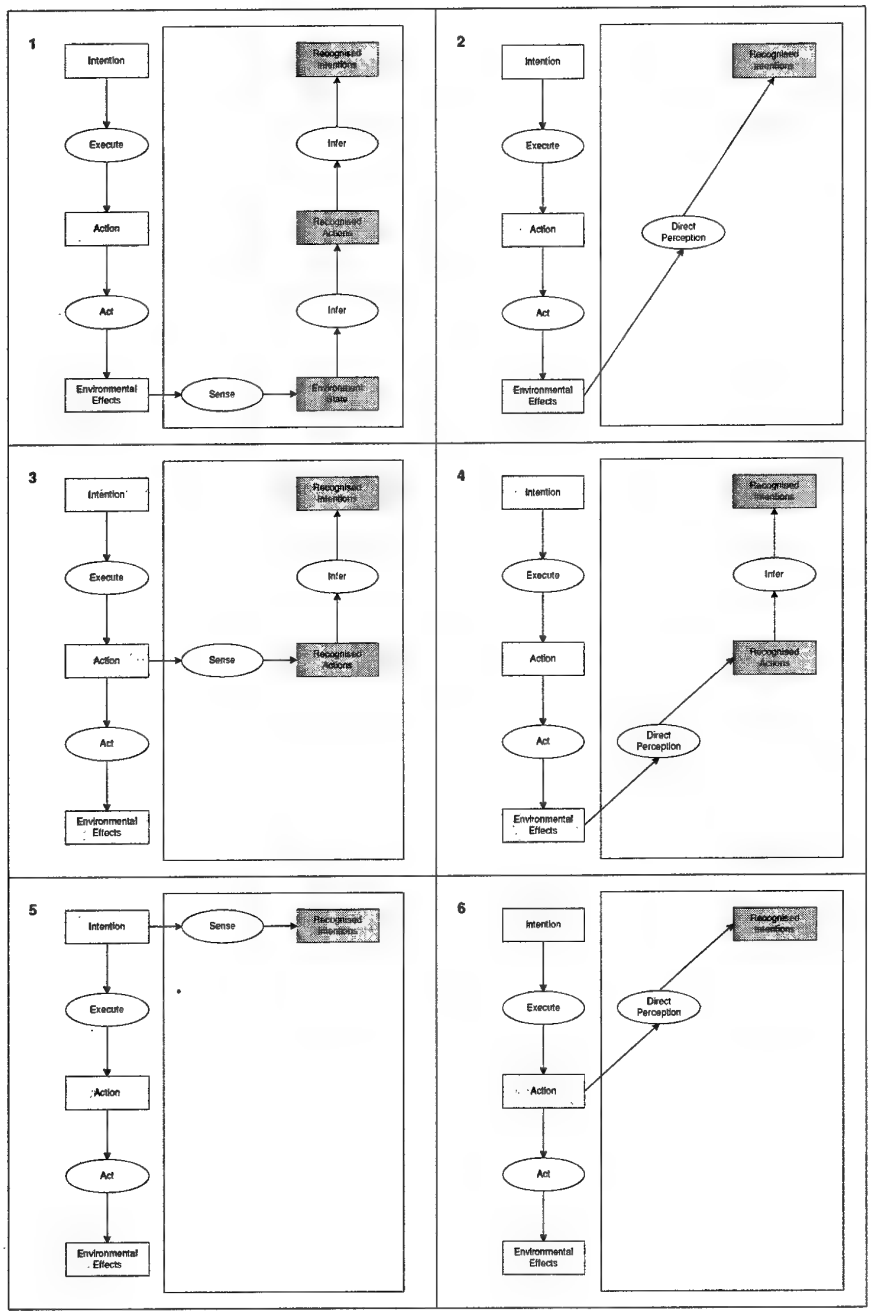

Figure 5.3: Many options become are available in the provision of intention recognition. Direct perception allows intent to be perceived directly without the need for the inference. Labelling the intending agent with either its actions, or intentions, allows sensing to be performed at higher levels of abstraction. This simplifies the intention recognition process still further. These options are only apparent when the environment is considered within the design process and when the modelling of perception as a sophisticated process capable of directly accessing higher order properties in the sensed data.

of intention recognition. These representations might be added later to the agent for the specific purpose of supporting intention recognition.

Fidelity of Cognitive Model In some applications the agent must implement a cognitive model. In these applications the agent must mimic the human reasoning process as it is expressed by the particular choice of cognitive model. The choice of cognitive model in these applications can constrain or influence the choice intention recognition process.

Choice of agent architecture and language With a dearth of commercial quality agent development environments and languages it is likely that an agent development will commit to a delivery architecture and language prior to a design. Though clearly not advisable from a standard software development perspective the novelty of agent technology and the time required to develop expertise in the application of agent languages means that the choice of language might well be made prior to design. This commitment to a language might also act to constrain the intention recognition design space.

Detail of Agent Reasoning During the agent design it must be determines how much detail the agent requires in its representation of intentions of the other agents. If, for example, simply knowing the general class of intention is adequate then the intention recognition process might be simplified. If the agent requires detailed knowledge of the expected possible future actions then the intention recognition process must supply this knowledge. This is comparable with the difference in object recognition between recognising the general class of an object and being required to reason about the many possible attributes of that object.

Performance Intention recognition as it has been applied in the past is often one of the more computationally expensive activities that an agent can undertake[46]. In the majority of systems performance is an issue. Shortcuts that can be achieved, particularly with respect to intention recognition are important considerations.

Ontologies If an ontology is mandated then the concepts that an agent must use are constrained. This will limit the scope for designing intention recognition.

Available Technologies It has already been suggested that modelling intention recognition with the agent's perception module is suggestive of technologies different to those that might be considered for inclusion within the agent's reasoning. In either case, be it pattern matching or inferential reasoning the technology necessary to perform the required task may well limit the possible designs. In dynamic real-time environments intention recognition can be both complex in design and time consuming in execution.

Encapsulation, Autonomy and Coupling The allowable coupling or the required autonomy may influence the selection of the design. Design options notably three, five and six have some degree of design coupling between the Intending Agent and the Recognising Agent.

[46]The technology demonstrator described in Section 2.3.3 devotes more time to intention recognition than it does to reasoning for action. This has the effect of more than doubling the execution time of an agent.

These constraints and influences on the design of the system must be taken into account early in the design process as they allow the short circuiting of many of the decisions. The agent ontology acts as the specification of the information required by the agent. These considerations can be mapped to the qualities of the designs. In the following chapter these constraints on design are presented as *problems* within the description of software patterns.

Chapter 6

Design Patterns for Modelling Intention Recognition

> *"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."*—Christopher Alexander et. al. [1]

Design patterns are the software engineer's way of documenting successful solutions to problems likely to be encountered again and again. Design patterns are primarily aimed at improving productivity by increasing reuse, but they also play a valuable role in documenting, maintaining, and extending existing systems [53]. Agent systems are proliferating and emerging patterns, programming idioms, shared ontologies, platforms and architectures that support design level-reuse assist in the maturation of agent oriented programming into a mainstream branch of software engineering. This chapter presents a set of design patterns that can be used and reused for modelling intention recognition in agent systems. The set of patterns presented has features of a pattern language. The software design patterns community uses the term *pattern language* for a set of related patterns that describe approaches to solve similar problems. Pattern languages form the basis of a way of talking about solutions to software design problems in a well understood shorthand that succinctly captures the salient aspects of a solution [53, 18]. Presenting patterns in this way allows the designer to consider and conceptualise possible solutions using the vocabulary defined within the pattern descriptions. This extends the utility of the patterns by allowing for combinations, comparisons, and trade-offs to be more easily discussed. Presenting agent system architectures as patterns provides an appropriate and familiar documentation style to facilitate their adoption by the wider software engineering community.

Section 6.1 provides necessary background prior to the elaboration of the patterns themselves. In Section 6.2 six patterns are presented corresponding to the six architectures that were a result of Chapter 5. For each pattern an actual example from a military air-combat simulation illustrates it use. The examples are extended further in a related domain and become the basis for the system descriptions of the flight simulator presented in Chapter 7. The language and style of Section 6.2 is different from other sections of

this thesis. This is deliberate to more closely resemble the published software patterns literature.

Section 6.3 provides a summary of the patterns and a guide to their application that allows the software engineering practitioner to more easily match problems to solutions. It is envisaged that the practical employment of agent design patterns will require application guides to manage the complexity of the large numbers of closely related patterns required to cater for agent systems development. Agents are more abstract and usually encapsulate greater functionality than the comparatively simpler *object*. Patterns will be required that can plug and play with other patterns and guides that lead the software engineer toward suitable solutions will be required. Even in object oriented development *meta-patterns* [143] and *pattern languages*[47] are moving in this direction.

6.1 Preliminaries

Almost all of the published patterns literature relates to object oriented analysis and design. What little there is was summarised in Section 2.6.1.1. Object-oriented design patterns are normally described with the aid of a template [53]. Templates standardise the pattern description thereby facilitating pattern consideration through familiarity with the presentation format. Section 6.1.1 outlines the pattern description template chosen for this thesis. Pattern description templates provide standard ways of documenting patterns. Although there are clear similarities between published agent patterns there are no documentation standards and the approach taken is to adapt the state of the practice of the object oriented patterns community. Section 6.1.2 describes briefly a possible categorisation of these patterns. Object oriented patterns are typically classified as architectural, functional, behavioural. etc. Section 6.1.4 describes the particular example chosen for illustrating the implementation of the patterns that follow in Section 6.2.

6.1.1 Agent Pattern Description

When presenting design patterns it is commonplace to define *templates* that link the patterns together in a consistent fashion [18]. With few published attempts to define templates or languages for agent patterns there is little in the way of precedent to build upon. Standards for the description and categorisation of agent patterns will emerge over time. Until then informed ad-hoc choices are the only alternative.

Deugo et. al. use a slight variation on the basic standard OO software engineering presentation format to describe patterns for communication in mobile agents. They specify the problem, the context, the forces, the solution and, finally, the resulting context [41]. In a similar fashion Kendall et. al. describe the problem, forces, and then the solution [95]. The Tropos project is another source of published agent patterns but again there is no definitive recommendation about an appropriate description template [24].

[47]Perhaps the best source of information about Pattern Languages are the proceedings of the Pattern Languages of Programming Conferences (PLoP). These are accessible on-line http://st-www.cs.uiuc.edu/plop.

The intention recognition patterns presented here demand a level of documentation that requires a sophisticated description template. The description style chosen is the one recently published by Buschman et. al. [18]. Similar to that of the Gang of Four [53] there are a number of differences, notably the addition of an *example* that extends through the description[48].

The chosen template is detailed and comprehensive enough to describe the subtle differences between the intention recognition patterns. A more detailed description of the origins, use and the practical application of patterns is available [53, 18] or in Section 2.6.1. Each pattern presented in Section 6.2 is documented textually and graphically with the following fields[49]:

Classification and Name The classification and the name should provide a memorable description of the pattern. A good name can form the basis of a pattern language and becomes the reference to the pattern when it is used in the context of broader descriptions.

Example The example provides a description of an actual problem that demonstrates the need for the pattern. Throughout the description of the pattern the example provides a grounded reference that keeps the discussion of the pattern practical and useful.

Context The context is a statement of the conditions under which the designer might consider using this pattern. For a given context there may be more than one applicable pattern leaving the designer with a choice.

Problem The problem describes the issues, problems, and challenges that the pattern is designed to address.

Solution The basic principal that defines the pattern that solves the problem. This is a description of the general nature of the solution and any relevant background information.

Structure The structure of the pattern is presented graphically to assist understanding of the functioning of the pattern. The OO community commonly adopts the UML to present a structural description of patterns and is an emerging standard for presenting system architectures [79]. Given the increasing interest that the UML is receiving from the agents community[50] the UML is also adopted here to present the agent architectures.

Dynamics The interactions between elements of the pattern (also known in the patterns literature as participants) are described and provide the basis designing interfaces and partitioning functionality. The dynamics of the high-level architectural design patterns in this Chapter will vary greatly with specific implementation. Although the UML's interaction, state, and collaboration diagrams, might be used to show the

[48]Variations on the Gang of Four style of presentation has also been adopted for the presentation of agent patterns by Findler et. al. in their work on patterns for social structures [111].

[49]These fields are exactly those of Buschmann et. al. [18]

[50]Not the least of which is a proposal under the auspices of the Object Management Group (OMG) to create an Agent UML (AUML) [132]

dynamic behaviour of agent patterns for the intention recognition patterns presented in this chapter annotations of the structure diagram are more appropriate. For this reason the Dynamics and the Structure of the intention recognition patterns are presented together and displayed in the same figure.

Implementation The implementation captures any known features, tips, tricks, traps, hints and guidelines that will facilitate the adoption and use of the pattern. Although this field is an important part of most pattern descriptions it is omitted. For agent systems the variety of possible implementations renders this field of little use other than as a summary of the important lessons from case studies. The reader is referred to Chapter 7 for examples of the implementation of these patterns.

Variants Any notable variants of the basic pattern are described.

Known Uses If there are any examples of the patterns in use they are described here. This allows the user of the pattern greater practical insight into the nature of the pattern and its employment. Any occurrence of the intention recognition patterns described in this chapter *in the wild* is the result of the post-hoc attribution of the pattern to the design and not the deliberate adoption of the design pattern as part of the system design. The patterns are presented as an attempt to map the design space and to characterise existing systems. As a result the known uses field describes systems that might be characterised as conforming to the architecture of the pattern. For some of the patterns this field is deleted as there are no systems (other than those described elsewhere within this thesis) that are known to conform.

Consequences Adopting any pattern will result in particular benefits and almost certainly some liabilities. These are documented to provide the engineer insights into the trade-offs that are made when adopting the pattern.

See Also This field describes relationships to other patterns, either as alternatives or dependencies. For most of the patterns this is neglected as entries becomes tightly self referential within the context of this set of patterns. Rather a 'relationship map' is provided in Section 6.3 that summarises the usage of the patterns and the relationships between them.

For agent patterns that deal with social interactions in multi-agent systems it may well be necessary to augment the template chosen, but it is adequate for describing agent architectural patterns of the type presented here.

6.1.2 Agent Pattern Categories

The object-oriented community *classifies* their software patterns into categories that reflect a broad grouping based on function. Typical categories include: structural, for patterns that describe data or process structures; creational, for patterns that describe solutions for creating object instances in different circumstances; and behavioural, for patterns that define the way an object operates. Although Lind proposes a useful set of agent pattern categories [108] there is no accepted categorisation for agent patterns.

There are so few recorded examples of agent patterns that any categorisation would be little more than a proposal.

Buschmann [18] describes three layers of patterns: architectural patterns; design patterns; and programming idioms. This categorisation is not quite so useful for agent systems. The differences between agent languages render programming idioms less relevant and the increase in the level of abstraction requires categories in addition to design and architectural. It has been a fundamental claim of this thesis that there is an important design interplay between the agent and the environment that influences both the system architecture and the internal design of the agent. For this reason the patterns here, although they are described as architectural patterns, also capture important aspects of the internal agent design.

The patterns presented here are *agent architectural*—each pattern describes a solution to the problem of implementing intention recognition in an agent system by presenting an architecture. In agent systems the architectural design is critical and goes to the heart of the manner in which the agent is situated in the environment and the manner in which agents collaborate. The pattern description includes details of both the relationship of the agent to the environment and to other agents and any relevant details of the internal design of the agent.

Other possible agent pattern categorisations might include: behavioural, describing some aspect of the operation of an agent; cognitive, describing the reasoning architecture of agents; social, describing the interactions between agents; and mobility, describing some aspect of the agents capacity to move over the web. Whether any or all of these classifications prove useful will only be known as agent technologies mature and become mainstreamed and there is a corresponding increase in documented experience with implemented agent systems.

6.1.3 Agent Pattern Evaluation

The design patterns are evaluated against seventeen criteria. These criteria are arranged under four headings. The first three categories reflect best practice in specifying agent architectures through their similarity with these IEEE standards [51]. A fourth category of evaluation criteria is needed to account for the cognitive modelling aspects related to agents and intention recognition.

1. *Module Criteria* where the important criteria are those assess the particular detail design requirements of the various modules and the relationships between them. Example criteria include:

   - complexity of the individual modules—specifically perception and reasoning;
   - appropriateness of the functionality assigned to a module;

[51] IEEE Std 1471-2000 "Recommended Practice for Architectural Description of Software-Intensive Systems" and IEEE Std 1016-1998 "Recommended Practice for Software Design Descriptions" suggest three system decompositions to specify an architecture. A difference is that the IEEE standards calls for a process decomposition but this is replaced (perhaps even subsumed) by a *non-functional requirements* category.

- cohesion of the module;
- coupling between agents;
- likely complexity of the implementation technology;
- implications for the total system.

2. *Data Criteria* considers the data available in the agent system to support particular architectures. This is particularly useful if large parts of the system are in existence. This view considers issues such as:

   - data that is provided to the perception modules from the remainder of the system;
   - the data that is required by/provided to the agent's reasoning processes;
   - the amount of data processing that is required.

3. The *Process Criteria* considers performance and complexity related issues that often impact upon the selection of an appropriate architecture.

   - the likely computational performance required of the system;
   - the resultant architectural complexity;
   - the total system complexity—taking into account architecture, technology, and module interactions.

4. Finally, and importantly, the *Cognitive Modelling Criteria* gives weight to issues associated with computational cognitive modelling. Only relevant for intelligent agent systems where computational modelling of intelligence is an issue this view consider such things as:

   - the division between perception and reasoning (cognition), specifically how much reasoning is performed by the agent reasoning;
   - the nature of the cognitive model; and
   - the location (in the various modules) of intention recognition.

Generally the software engineering process will consider many, if not all, of these categories when selecting the appropriate pattern and no one will dominate.

6.1.4 Illustrative Examples Used Throughout this Chapter

Exposing the detail of the implementation of patterns requires substantive illustrative examples [18]. Notwithstanding the detailed description of the practical application of these patterns that follows in Chapter 7 it is useful to provide a brief but hopefully enlightening example for each patterns that provides insight into the relationship between the problem that the pattern solves and the implementation detail of that solution.

The examples provided for each of the patterns arises from military simulation. This domain was chosen because: there are strong examples of each of these patterns available;

the implementation details and design are well known to the author (much of the design and development of these systems was undertaken by the author whilst employed by the Defence Science and Technology Organisation); some details have been published in the literature [169, 78, 76]; and the specific requirements of this domain fostered the research that led directly to this thesis.

It is important to note that the development of the simulation systems described in these examples preceded the description of the design patterns. The designs chosen were as a result of a software design process that did not consider the application of patterns, suitable patterns did not exist. Nor did the development of these simulations give rise to the patterns directly. The post hoc attribution of the patterns to various incarnations of these military simulations was a fortuitous result of an examination of the simulator architectures following the documentation of the intention recognition patterns. It is possible, even likely, that the experiences with simulation design unconsciously influenced the development of the design patterns. If so the influence was entirely appropriate. Design patterns should be the result of experiences with successful software development.

Each of the examples consider architectural design decision that were made during the development of a military simulator. In these types of simulators intelligent agents are often the technology of choice for the implementation of computer generated forces (CGF). CGFs benefit from intention recognition because it allows them to be less easily seduced by human pilots who often quickly learn to *trick* predictable agents; because it improves coordinated behaviour by allowing the agent to predict the future actions of its team members; and because it allows the agent to more accurately reflect the intention recognition performed by real fighter pilots.

These simulators are highly complex software systems that have been built-up over a number of years. In most cases the intelligent agents within the simulation have been added as an afterthought in response to available technology and were not an original component of the system. Major alterations to the system are often prohibitively expensive and agent-design compromises often result. In the examples that follow it is appropriate to visualise the agents as *surrogate pilots* flying as computer generated adversaries inside the simulator. Implementing intention recognition in these agents is conditioned by the constraints and requirements that help to specify the system.

6.2 Agent Patterns for Intention Recognition

This section presents six agent architectural patterns that might be used to assist the modelling and design of intention recognition for intelligent agent systems. The six patterns are the set of possible variations available when the intentional activity of an agent is characterised on three levels: the intentional level; the activity level; and the state level; and recognition is conducted at those three levels. Details of the model that led to these patterns is presented in Chapter 5.

The patterns described here at the architectural design level, and although they may point toward certain implementation technologies the gap between abstract agent architecture and implementation is likely to be wider than is the case for object-oriented systems. Attempts have been made to canvass implementation possibilities for each of the designs

and these are documented under *implementation*. Figure 6.1 shows a UML use case diagram that depicts the relationship between these patterns[52]. These types of diagrams are useful for expressing the relationships between patterns and the manner in which they relate to the requirements but should not be confused use case patterns [53].

The patterns here share a number of common elements (or participants as they are often referred in the patterns literature). A brief description of these participants is included here to avoid duplication in the pattern descriptions that follow.

Recognising Agent The agent that requires the capability to recognise the intentions of other agents. This agent conforms to the definitions presented elsewhere in the thesis (Section 2.1) and has perception, reasoning, and action sub-components.

Intending Agent An agent who's intention will be recognised. This is not to imply that it actually needs to manifest some representation of intention as it executes. Intention can be ascribes by the recognising agent.

Labeller The labeller is a system component responsible for labelling an agent with an indication of its internal state. In practice the labeller can be implemented in such a wide variety of ways that the term *labeller* might be a bit misleading. For example the labeller might be as simple as an accessor function that an agent to read the internal state of the agent. The Labeller might be a function internal to the Intending Agent that actually does provide an externally readable *label* that can be accessed by other agents. Even in this case there are many options. The labeller might need to provide a different label for every other agent in the system. Or perhaps a number of labels that differ on context. So that an agent reading the label will see a different label depending upon its situation. Perhaps the labeller is a function that is external to both the Intending Agent and the Recognising Agent, residing somewhere in the environment in a dedicated *Intention Recognition* module that handles all of the intention recognition modelling for the system. The primary distinguishing feature of this participant is that it presents an indication of intention or action to the recognising agent thus removing the need for any processing internal to the agent. The particular design chosen will be strongly influenced by the design of the intending agent.

Sensor The Sensor is a Perception component that gathers data from the environment and provides it to the reset of the system. It manipulates the data in a trivial fashion, minor transformations of the data are allowed but aggregation, fusion, and abstraction of the data are outside the function of this module.

Action Inferencer The action inferencer is a system component responsible for recognising actions in the state data. The module is internal to the reasoning of the recognising agent.

Intention Inferencer The intention inferencer is a system component responsible for recognising intentions in the action data.

[52]The notation for the agent (an actor inside a circle with an arrow) is an extension to UML originally proposed by Papasimeon and reported by Heinze [75]. This notation is not standard and is introduced here but is not used in the pattern descriptions in the interests of conforming to agent standards.

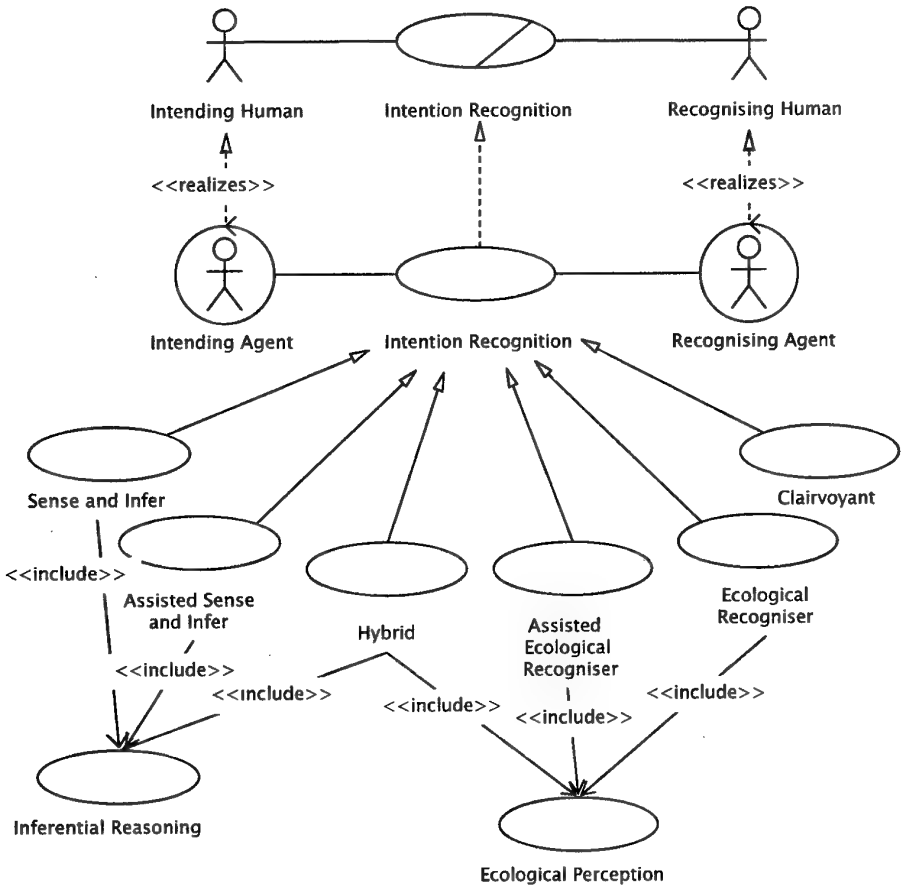

Figure 6.1: Use Case Diagram for Intention Recognition Patterns. This diagram shows the relationship between the two human participants involved in intention recognition captured by the intention recognition business case (the business case is designated by the diagonal stripe). This real-life *case is* realized *by the agent use case below it. The design patterns are variations on the abstract intention recognition use case. Finally the use cases include two further use cases* Inferential Reasoning *and* Ecological Perception.

Pattern Matcher The pattern matcher is a system component responsible for recognising intentions in the patterns of input data. Although the function might be considered similar to the Inferencer components there are two significant differences: the pattern matcher is a *perception* component and therefore isolated from the agent reasoning; the pattern matcher is a self contained and autonomous component of the agent responsible for data abstraction. The participant is names *Pattern Matcher* but ultimately any technology capable of mapping from physical state to higher order representations will be adequate. Obviously the particular properties of the system will dictate the type of solution required.

Environment The environment is a participant responsible for storing the physical state of all entities in the system and for modelling the interactions between them.

These participants are the base components used in the architectural pattern descriptions that follow.

6.2.1 Pattern: Hybrid Recogniser

This pattern implements a hybrid solution by combining aspects of the ecological family of patterns and aspects of the sense-and-infer patterns. The Hybrid Intention Recognition agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent perceives directly the activities of the Intending Agent in the data sensed from the environment and then infers its intent. This pattern gains the name Hybrid Recogniser because it is a combination of the Ecological Recogniser and the Sense and Infer patterns.

6.2.1.1 Example

An air combat simulation requires computer generated forces that can infer the intention of others. A major difficulty is in designing the means by which the agents determine what particular tactics the aircraft in the simulation are adopting. Once that is known it is a relatively simpler task to infer the intentions. If an agent knows that an adversary aircraft has completed a particular manoeuvre and has just commenced a new manoeuvre then it is straightforward to string those building blocks together into an integrated intentional view of the situation and to predict what will follow.

6.2.1.2 Context

The provision of intention recognition in an intelligent agent system.

6.2.1.3 Problem

When intention recognition is required and the features of the problem domain are similar to those of the *Ecological Perceiver* or *Sense and Infer* then the Hybrid Recogniser offers a viable alternative. This pattern is particularly relevant if the mapping between the

environment state and the recognised activities is complex but does not depend upon the Recognising Agent's mental state whereas the mapping between the recognised activities and the recognised intentions is simpler but is influenced by the mental state of the agent. This pattern provides a compromise between the completely inferential approach of the Inferencer Pattern and the completely perceptual approach of the Ecological Intention Recognition Pattern.

The Hybrid Recogniser pattern should be used under the following conditions:

- The system outside of the agent that is conducting the recognition cannot be designed or modified to support intention recognition.
- There is a well defined boundary that separates agent perception and reasoning.
- The mapping from actions to intentions is simpler than from environment state to actions.
- The mapping from the environment state to actions is relatively uninfluenced by the agent's mental state unlike the mapping between the resulting actions and intentions.

6.2.1.4 Solution

The solution is to provide a Pattern Matcher to map state data onto action data and then use the Intention Inferencer to reason about the intentions. This solutions combines two quite dissimilar approaches, hence the name Hybrid Intention Recognition. The choice of pattern matcher for the first stage and inferential reasoning for the second is appropriate from a computational sense in managing the types of data and cognitively in psychological sense in the manner in which it maps to theories of human psychology.

6.2.1.5 Structure and Dynamics

The structure and dynamics of the architecture is described by Figure 6.2.

6.2.1.6 Example Resolved

The implementation of this example was the first stage of a program to transfer the content of this thesis to military simulation. CLARET was used as a pattern matcher and passed descriptions of recognised manoeuvres to the dMARS agents that were used to model the fighter pilot reasoning. These manoeuvre descriptions where then used as the building blocks of an inferential reasoning process that resulted in intention recognition. The system proved capable of recognising a wide range of intentions if the scenarios were relatively simple. As the number of simulated aircraft was increased beyond the single-opponent case ambiguity in identity proved problematic.

6.2.1.7 Variants

The variants of the Hybrid pattern are the possible combinations of the variants of the Sense and Infer pattern and the Ecological Recogniser pattern.

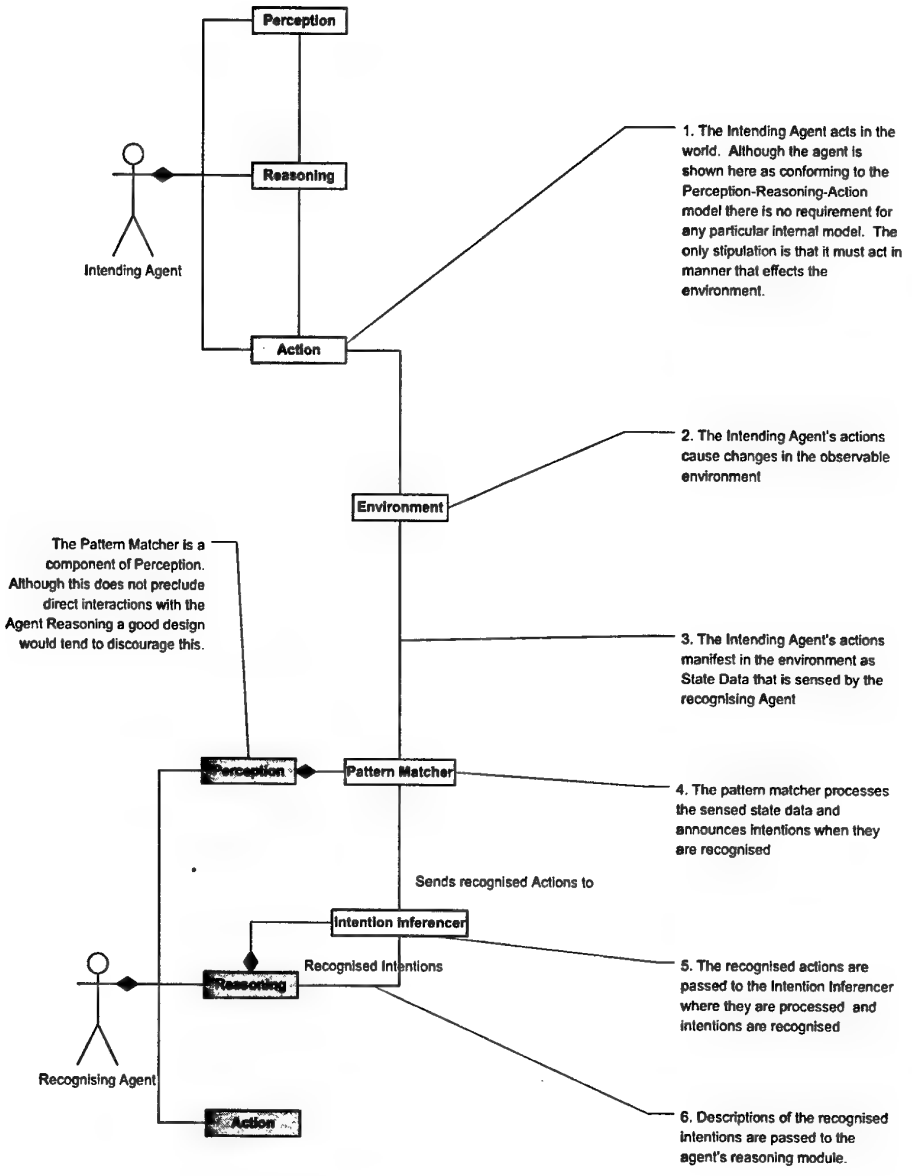

Figure 6.2: Pattern Structure: Hybrid. The Hybrid pattern makes use of state data from the environment which is passed to a perception module. The pattern matcher processes this data in search of actions. *As these are recognised descriptions of thenm are passed to the agent's reasoning module where they are used to infer the existence of intention.*

6.2.1.8 Consequences

Some of the benefits of the Hybrid Intention Recognition Pattern:

1. **Autonomy** By maintaining the intention recognition capability entirely within the agent and requiring no modifications to other components of the system the agent is kept as a completely autonomous entity. It is designed for a particular environment, of course, but changes to the internals of any other system component will not effect the agent. The agent is completely responsible for the intention recognition process and requires no external assistance.

2. **Cognitively Interesting** This pattern provides the architecture that comes closest to mainstream views about the nature of human cognition. The recognition of activities is modelled as a component of perception but relating those activities to each other and inferring more abstract properties like intention is assigned to the agent's reasoning. This draws the line between perception and cognition at a midpoint of the ongoing AI debate about the nature of models of human intelligence [26].

and the liabilities:

1. **Complexity** The implementation will require both inferential reasoning and pattern matching. Though not mandating different languages or technologies, more sophisticated implementations may require this. Agents composed from hybrid technologies are often afflicted with greater complexity in design testing, validation and maintenance and all of the problems of hybrid technology solutions. Not only is the agent multi-technology but the provision of intention recognition is multi-technology. The agent developer is likely to make use of two distinctly different technologies in the provision of the required functionality.

6.2.1.9 See Also

The Hybrid pattern, as the name suggests, is a hybridisation of the Ecological Recogniser and the Sense and Infer patterns.

6.2.2 Pattern: Sense-and-Infer

This pattern is named for the dominant theory of visual perception that is reflected in its architecture. The agent will recognise intentions by sensing the environment and inferring higher order concepts. The Sense and Infer agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent senses the environment state and infers the intention of the Intending Agent in a two step process.

6.2.2.1 Example

Adding an intention recognition capability to the existing computer generated forces in a military simulator gives them the capacity to respond more quickly and intelligently to events that occur. One of the clearly identified requirements for these types of simulators is intention recognition [115]. In order that these intelligent agents pose a realistic threat they are to be given the capacity to infer the intention of other pilots and to anticipate their future actions. This is useful for both adversarial situations where anticipating an opponent allows for preemptive action and in cooperative situations where anticipation smoothes collaboration and reduces the need for communication.

The simulation system is in operational use and the intention recognition capability must be added to it. The agents as they currently exist make use of a high-level agent language and implement very sophisticated behavioural functionality [74]. The behavioural repertoire of the agents is such that the agents must integrate intention recognition very tightly so that recognition can be influenced by the agent's current activities.

6.2.2.2 Context

The provision of intention recognition in an intelligent agent system.

6.2.2.3 Problem

Intention recognition is required in an agent system where there are some fundamental restrictions on the design: entities other than the Recognising Agent may not be modified to support intention recognition; and there is a requirement that intention recognition be integrated in a sophisticated way with other agent reasoning behaviour. The first requirement might occur when there are legacy issues, the agent is being designed to be placed into some third party system or the cost of modifying other parts of the system to support intention recognition is prohibitive.

Use the Sense-and-Infer architecture to balance the following forces:

- The environment and other agents cannot be modified to assist the Recognising Agent with the intention recognition task. This precludes direct access into the internal state of other agents is not possible. This is commonly the case if the intention to be recognised is that of a human interacting with the system[53]. In a system inhabited only by agents this suggests that there is no direct contact allowed between the agents supporting intention recognition. All contact is indirect and sensed through the environment.
- It is desirable that there be low coupling between the agent and the rest of the system. This is a common requirement in developing agents that can be reused

[53]Whilst direct access to the internal intentional state (if such a thing even exists) of a human is not possible, a human using a software system can be questioned about their intent, or their beliefs, goals, or plans. Leaving aside issues of the reliability of the response it is possible for an agent to gain some insight into the mental state of a human without resorting to inferential reasoning.

in multiple environments. If the agent system is open (there is a requirement for interoperability with other, as yet undefined systems), required to be modular, or unstable then low coupling between agents is desirable.

- The agent should implement a model of intention recognition that has some psychological plausibility. The Sense and Infer architecture implements places responsibility for intention recognition entirely within the Recognising Agent. This matches intuitions about the nature of human intelligence although from an AI perceptive it places relatively more functionality into cognition than perception.
- The intentions to be recognised are those of a human user of the system or there is no access to the internal mental states of other agents.
- The set of state data used to infer the intentions is relatively small. If the set of state data required for intention recognition is large then there may be a significant overhead incorporating this data into the agent's reasoning.
- Performance and complexity issues will not dominate. The Sense and Infer will result in a more complex implementation than other patterns. The performance of intention recognition will be determined largely by the specifics of the domain and the implementation but all other things being equal the Sense and Infer pattern is one of the more complex patterns.
- There is the capacity or requirement to reason at multiple levels of abstraction. If the agent must reason at multiple levels of abstraction for other purposes. Perhaps in fulfilling other reasoning requirements the agent makes use of the state data required for intention recognition then there is little disincentive to avoid state-data for intention recognition.
- The inferential reasoning required for the Sense and Infer pattern is likely to be complex. If there are other equally complex processes occurring within the agent then the overhead of this pattern may be *relatively* small.
- The recognised intentions are highly subjective or situation dependant. If the internal state of the Recognising Agent (particularly the state of its reasoning) influences the recognition process then the Sense and Infer pattern is strongly suggested.
- There are relatively few intentions to be recognised. If the scale of the recognition problem is small then the complexity of the pattern can be held at a manageable level.

6.2.2.4 Solution

Introduce a Sense-and-Infer architecture to provide intention recognition encapsulated entirely by the agent's reasoning. The agent first *senses* the data available in the environment and then infers intention in a two stage process. The first results in a representation of the recognised actions and the second step result in recognised intention.

By using the Sense-and-Infer pattern an agent is decoupled from the environment and applications support heterogeneous agents and more flexible reuse. The inference of intention capability is encapsulated by the agent creating a cleaner interface.

Inference of intention is conducted inside the agent and there need be no direct mapping between any *actual* intention and that which is inferred. Intention is *in the eye of the beholder*. Because the intention recognition is conducted entirely within the agent's reasoning module there is greater opportunity to exploit interactions between intention recognition and other agent behaviours.

6.2.2.5 Structure and Dynamics

The structure and dynamics of the Sense and Infer patterns is described by Figure 6.3.

6.2.2.6 Example Resolved

The Sense and Infer pattern was applied to the design of the computer generated forces. The agents sense the unmodified simulation environment through a constructed interface that models the senses and sensors available to the pilot. This sensory data is then processed by a module that *observes* actions as they occur. This observation classifies actions based on a set of rules that map aircraft trajectories and specific discrete events (such as an observed missile firings or radar detection events) onto descriptions of manoeuvres, tactics, and actions. The first inferencing process was implemented within the standard dMARS language using a set of *plans* to define the procedures by which manoeuvres and tactics were recognised. This set of observed manoeuvres, tactics and actions is passed to a component that conducts hypothesis based reactive recognition over a set of possible intentions. This was implemented with a custom extension to the dMARS language [19]. When only a single option remains then recognition component announces that it has successfully recognised the intention. Interactions between these two processes were permitted as were interactions with any other agent reasoning process. More details of simulation are available in Section 2.3 and of the theory of the reactive recognition algorithm used by Rao and Murray [151]. It should be noted that the development of this software pre-dated this research into patterns and so the assignment of the Sense and Infer pattern to this system, though perfectly reasonable, was post-hoc. This implementation added significantly to the processing overhead of the agents and although successfully demonstrated never become operational.

6.2.2.7 Variants

A number of variants of the basic Sense and Infer pattern are possible:

1. **Single Step Sense and Infer** In simpler systems the *infer* stage of the intention recognition process might be conducted in a single step. The agents reasoning might infer intention without requiring the intermediate stage.

2. **Multi-step Sense and Infer** Rather than *action* as the intermediate step there might be some other intermediate step that might be as useful. In more complex systems there may be several intermediate reasoning steps that are necessary along the way to a completing the intention recognition. Tidhar and Sonenberg [162]

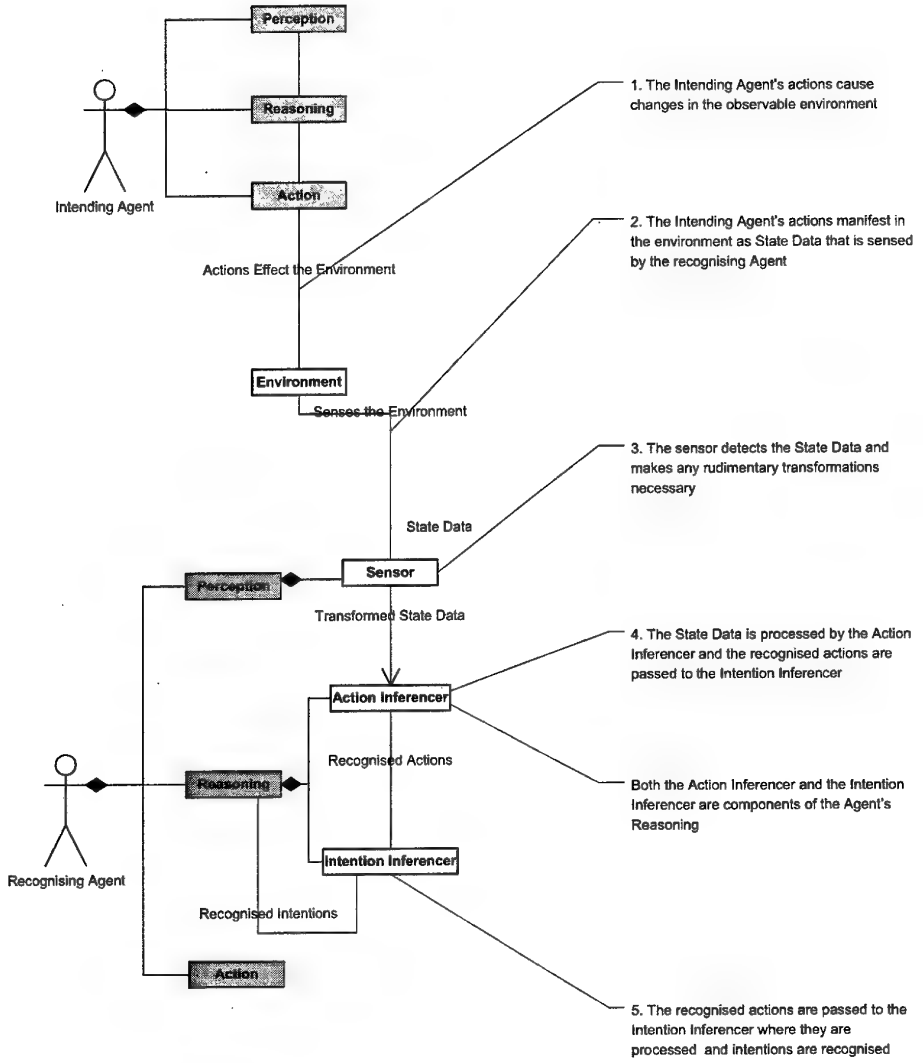

Figure 6.3: Pattern Structure: Sense and Infer. The internal detail of the Intending Agent *is inconsequential to the pattern as the* Recognising Agent *senses only changes in the environment state. In practice, execution will possibly involve asynchronous parallel processing at each step and not the synchronous sequence shown here but the process is similar. An agent, the* Intending Agent *acts in the world and the effects are sensed by the* Recognising Agent. *The Sensor is responsible for Sensing the appropriate state data in the environment and passing this information to the Action Inferencer. The Action Inferencer, utilising inferential reasoning or an equivalent, determines the actions taken by the agent that gave rise to the sensed state of the environment. The Recognising Agent assimilates the sensed state-data into its reasoning and through some reasoning process, characterised here as inference but it need not be, develops a subjective classification of the actions that led to that state-data. The Recognising Agent then process the classified actions to recognise the intentions that gave rise to the actions.*

propose six different steps, in a process that has the basic look and feel of the sense and infer pattern but deals with intention recognition in the complex social environment of multi-agent systems.

3. **Sense and Infer with Plan Recognition** Perhaps an obvious replacement intermediate step for conducting the inferential part of the pattern is to focus on *plans* and plan recognition rather than actions. The nature of intention can then be constructed from knowledge of the plans. For some researchers there is little difference between plan and intention recognition but adopting a view of intention such as that of Rao makes plans an obvious intermediate point for intention recognition.[54]

6.2.2.8 Consequences

Some of the benefits of the Sense-and-Infer pattern are:

1. **Autonomy** By maintaining the intention recognition capability entirely within the agent and requiring no modifications to other components of the system the agent is kept as a completely autonomous entity. It is designed for a particular environment, of course, but changes to the internals of any other system component will not effect the agent. The agent is completely responsible for the intention recognition process and requires no external assistance. In the mainstream terminology of software engineering autonomy could equally be referred to as encapsulation.

2. **Coupling** As a consequence of the autonomy of the agent and the lack of direct dependence upon other modules, coupling is low.

3. **Reuse** Because the agent is designed to infer intention from low level state data and is less coupled to other agents than in other patterns reuse across environments is encouraged.

4. **Agent Heterogeneity** A consequence of inference of intention operating exclusively with state data observed in the environment there are no special requirements placed on the internal design of other agents. This allow for agent systems where agents can recognise the intentions of any agents regardless of their internal structure.

5. **Cognitive Plausibility** The process of intention recognition defined by the Sense and Infer pattern is similar to mainstream theories of human perception and intention recognition.

6. **Intentional Ascription** Intention recognition with the Sense and Infer pattern requires no explicit representation of intention in the agents whose intention is being recognised. The intention recognition process is completely one of ascription by the recognising agent.

[54] I should comment on this further somewhere - maybe in the scope - or maybe in Chapter 5. This sees plan recognition as a precursor to intention recognition. Although this matches certain views of the subtle difference between plan and intention recognition there are many practical examples of applied plan and intention recognition that blur the distinction enough to render it meaningless.

7. **Integrated Reasoning** When the intention recognition process is part of the wider reasoning processes of the agent there is scope for modelling sophisticated behaviours. Behaviours where intention recognition is more directly influenced by and in turn influences other aspects of the agent behaviour.

but the pattern also suffers from four liabilities:

1. **Agent Complexity** The inferential reasoning required to recognise intentions of other agents can be highly complex. The actual complexity is of course domain dependant but the Sense and Infer pattern will generally result in a more complex and sophisticated agent structure.
2. **Ontological Purity** With the Sense and Infer pattern the agent reasons with the low-level state data sensed from the environment. This contradicts the general concept of an agent as a software entity that should manipulate more abstract concepts.
3. **Performance** The complexity that results from the Sense and Infer pattern can lead to performance problems.
4. **Practicality** The Sense and Infer pattern is impractical if the set of state data is large that must be reasoned with to infer actions and intention is large. The Ecological Recogniser pattern may be preferred in cases where the state-data set is large, likely to be modified, or subject to change with context.

6.2.2.9 See Also

The Assisted Sense and Infer pattern is the next closest relative of the Sense and Infer pattern but the two others most likely to meet the same set of requirements are the Hybrid Recogniser and the Ecological Recogniser.

Again the process, characterised here as inference, might be some other form of reasoning but the significant feature of this pattern is that at either of the processing is considered to be internal reasoning of the agent and not a part of the perception module. At any stage feedback or feed-forward between the two inferencing processes is viable allowing recognition of an intention to influence future recognition of actions etc. Furthermore, the recognition process can be influenced by any other agent activities. In this way the recognition process can be truly subjective. Not just subjective in the sense that it depends upon an agent's local view, but that it depends also upon the agents internal state. This is possible within this pattern because the entire recognition process is conducted as a part of the agents larger reasoning processes.

6.2.3 Pattern: Assisted Sense and Infer

This pattern is a variant of the sense and infer pattern. The recognising agent is assisted in its task by virtue of direct access to named representations of actions in the environment. The Assisted Sense and Infer agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent senses the activity state of the Intending Agent and then infers its intent.

6.2.3.1 Example

The air combat simulator described in Section 6.1.4 was proving difficult to maintain. While the number of possible manoeuvres, tactics, and actions was small the task of disambiguating them was tractable, but as the repertoire grew it became increasingly obvious that there were significant problems in maintaining the system. Furthermore there was a computational overhead that introduced a substantial performance problem. The difficulties were not with the second inferencing stage, that which maps actions to intentions but with the first, the mapping of state data into action. The problem occurred because, as the number of actions to be recognised grew so did the complexity of disambiguating the possibilities (See Heinze et. al. for a more detailed discussion of the problems of disambiguating tactics in air combat simulation). A design solution was needed to manage the complexity of the inferencing process but that required little modification of the underlying simulation infrastructure. Adding to the difficulties the agent implementation language was quite well suited to the second stage inferencing where the rules specifying the relationships between actions and intentions were quite simple but far less suited to the first stage where the rules where based primarily on complex spatio-temporal geometric relationships between aircraft.

6.2.3.2 Context

The provision of intention recognition in an intelligent agent system.

6.2.3.3 Problem

Intention recognition is required of agents in an agent system but there is some access available to the internal state of the agents (this is not normally the case with human users of a system). The accessible internal agent states indicate their current activities and these might be used to facilitate intention recognition. Beyond simple availability these states must also be suitable and useful for supporting intention recognition. It might be that access to the agents states is already available in some form or it might be that other agents can be specifically designed to manifest these states, or that some process generates them.

The Assisted Sense and Infer pattern should be used to under the following conditions:

- Representations of the activities of the agent is accessible or can be created.
- It is possible to design, modify, or extend the Intending Agents to provide the support for the provision of the activity state data.
- These representations are suitable for supporting intention recognition.
- It is acceptable to create a coupling between agents for the purposes of intention recognition.
- Inference of intention can process data at the activity level and above. There is no requirement that the intention recognition process reason about detail beneath the activity level. This suggests that disambiguating intentions or making use of the

knowledge about another's intention does not require access to data about intention at the environment state data level.

6.2.3.4 Solution

The Assisted Sense and Infer pattern is similar to the Sense and Infer pattern but uses direct access to the actions of the *Intending Agent* to remove the first inference step. The access to direct knowledge of the actions can be conceptualised as reading the *labels* that are placed on an agent but making knowledge of the actions of an agent directly available to other agents might be achieved in several ways but in most cases this will require some modification to system components other than the recognising agent. Once the actions of an agent are known an inference process can infer intention in much the same way as for the Sense and Infer pattern.

6.2.3.5 Structure and Dynamics

The structure and dynamics of the Assisted Sense and Infer pattern are shown in Figure 6.4.

6.2.3.6 Example Resolved

The Assisted Sense and Infer pattern was applied to provide the agent conducting the recognition with pre-determined representations of the required actions. Agents in the flight simulator make decisions that result in the selection of particular manoeuvres—ultimately as an enumerated type. Global access to these *manoeuvre numbers* was provided, removing any ambiguity, or need, for the first inferencing stage. The reactive recognition component described in Section 2.3.3 was then used, in a form unmodified from that described in the Sense and Infer pattern to infer intention based upon the knowledge of these manoeuvres. Although the implementation of this design simplified the agent reasoning it reduces the face validity of the simulation by adopting a somewhat artificial method of the Recognising Agent becoming aware of the manoeuvres of other agents. It also introduces an artificial coupling that ties the agents together for the single purpose of modelling intention recognition.

6.2.3.7 Variants

The variants of the Assisted Sense and Infer pattern are dictated by the type of data that is made available to the Recognising Agent. A representation of the activities and actions of the Intending Agent was made available in the description of this pattern but alternatives might choose some other equally useful representation. Alternates would include those features that are related to activities: *plan* or *function*; and those that are either more declarative in nature such as *belief* or the state of a database or the result of a query.

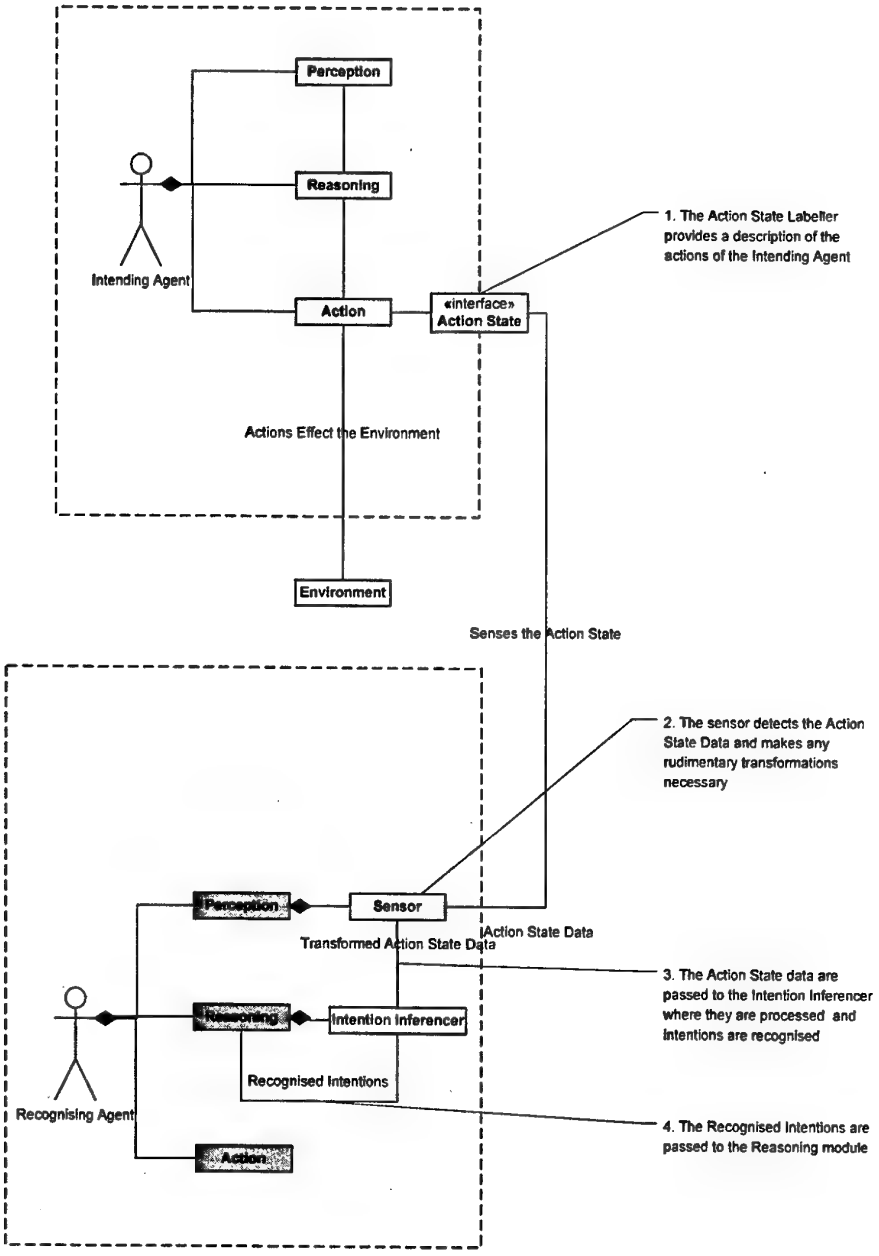

Figure 6.4: Pattern Structure: Assisted Sense and Infer. It is essentially a simplification of the Sense and Infer pattern through the mechanism of access to the action state of the intending agent.

6.2.3.8 Consequences

The benefits of the Assisted Sense and Infer pattern are:

1. **Simplicity** As compared with the Sense and Infer pattern the Assisted Sense and infer is structurally and computationally simpler. The provision of *action-level data* directly allows the first, and usually the most complex part of the two inference process of the Sense and Infer pattern to be removed.

2. **Cognitive Plausibility** The Assisted Sense and Infer pattern implements a general model of intention recognition that maintains some cognitive plausibility. Short circuiting the inference step that transforms state-data into inferred actions sacrifices little in the way of cognitive plausibility.

3. **Intentional Ascription** Intention recognition with the Sense and Infer pattern requires no explicit representation of intention in the agents whose intention is being recognised. The intention recognition process is completely one of ascription by the recognising agent. Knowledge of the actions of the other agent is provided directly but most agent types will have some representation of their actions whereas few maintain an internal representation of intent.

4. **Integrated Reasoning** When the intention recognition process is part of the wider reasoning processes of the agent there is scope for modelling sophisticated behaviours. Behaviours where intention recognition is more directly influenced by and in turn influences other aspects of the agent behaviour.

5. **Natural Solution** The assisted Sense and Infer pattern uses labelling for the more objective part and inference for the more subjective part. This results in a more *natural* solution and is a compromise between the extremes of Sense and Infer and Clairvoyant but without the need for pattern matching seen in Ecological and Assisted Ecological..

but the pattern also suffers from liabilities:

Coupling The Assisted Sense and Infer pattern introduces a dependency between the designed actions of an agent and the intention recognition process of the agent performing the recognition. This coupling, though present, is not as severe as in the Clairvoyant pattern.

Performance Though a simplification of the Sense and Infer pattern the Assisted Sense and Infer pattern will still be relatively computationally complex.

6.2.3.9 See Also

The two most closely related patterns to the Assisted Sense and Infer are its more complicated predecessor, the Sense and Infer pattern, and the Assisted Ecological Recogniser that adopts a similar approach to granting access to the activity state of the intending agent.

6.2.4 Pattern: Ecological Recogniser

This pattern is named for the theories of ecological visual perception upon which it is based. The Ecological Recogniser agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent perceives directly the intention of the Intending Agent *in the patterns of data in the environment. This pattern is named the Ecological Recogniser because it is inspired by the fundamental principles of theories of ecological visual perception [57].*

6.2.4.1 Example

Recognising and disambiguating the tactics employed by an adversary in a military context is not a simple matter. Situational dynamism, uncertain knowledge and identification problems combine to create an environment where intention recognition is important but difficult. A fundamental requirement was that the intention of another entity was determined prior to the completion of the intention to allow for preemptive response. Even if unfettered access to the actions of other agents is available inferring their intention can still be intractably complex. When the air combat simulator described in Sections 6.1.4 was expanded further the scenarios of interest began to include more aircraft. This in turn created even more ambiguities in recognising and intention and the designs described by the Sense and infer and Assisted Sense and Infer patterns became unworkable. A solution was required that reduced the design and program complexity without sacrificing too much in the way of cognitive plausibility. The main impediment to the continued use of inferential reasoning as a means of determining intention was associated with the large design and programming effort required to cater for the many possible cases, with all their subtle variations, in a robust reliable manner. There seemed little that could be done to simplify the problem, it was inherently complex, but confining the solution to the agent reasoning was unnecessarily complicating the agent design. By now about one half of the agent was devoted to intention recognition.

6.2.4.2 Context

The provision of intention recognition in an intelligent agent system.

6.2.4.3 Problem

An intelligent agent system must recognise the intention of other agents but has access only to the state data. Furthermore the state data is complex and the intentions to be recognised are difficult to describe by sets if rules.

The Ecological Intention Recognition pattern should be used under the following conditions:

- no need to reason about detail of intention - a simple announcement of the type os intention is adequate

- there is no access to data other than the state data
- mapping the state data to the recognised intentions is complex.
- there is a lot of state data
- the agent should reason with high level representations of intention
- definitions about exactly what constitutes a specific intention are difficulty to codify in rules but relatively easier to provide in the form of examples.

6.2.4.4 Solution

Implement a recognition module within the agent but considered a part of perception that pattern matches the incoming data streams onto learned, stored, or otherwise supplied examples of the recognised intention.

6.2.4.5 Structure and Dynamics

The basic structure of the Ecological Recogniser Pattern is very simple but this is achieved at the expense of encapsulating a great deal of complexity inside the pattern matcher. This hides the detail of intention recognition from the reasoning of the agent and provides the complete mapping from state data to intention.

The dynamic behavior of this pattern depends very much upon the particular implementation technology but in general the pattern matcher takes a data feed from the environment of all of the necessary data and maps this into the required descriptions of intent. This makes the processing from an architectural perspective almost trivial. Internally the Pattern Matcher is expected to recognise intentions in the data available in the environment.

The operation of the Pattern Matcher will depend upon the specific of the chosen solution. In Section 6.2.4 an example this technology at work is provided.

6.2.4.6 Example Resolved

The computer generated entities in an air-combat simulator were fitted with a pattern matching algorithm that provided the capacity for intention recognition to operate over a number of adversaries as the simulation unfolded.

This allowed intention recognition to be trained by examples rather than encoded in rules. The subtleties of disambiguating particular cases are lost in the algorithm inside the pattern matcher and appropriately selected training sets. Recognised intentions are announced to the agent by the pattern matcher simplifying the agent design. For a description of the use of this pattern matching technology applied to air combat simulation see Heinze et. al. [72] and Pearce et. al. [139].

The pattern matching algorithm was fed with data streams from the simulation environment and matched the observed traces against known examples. Successful matches are

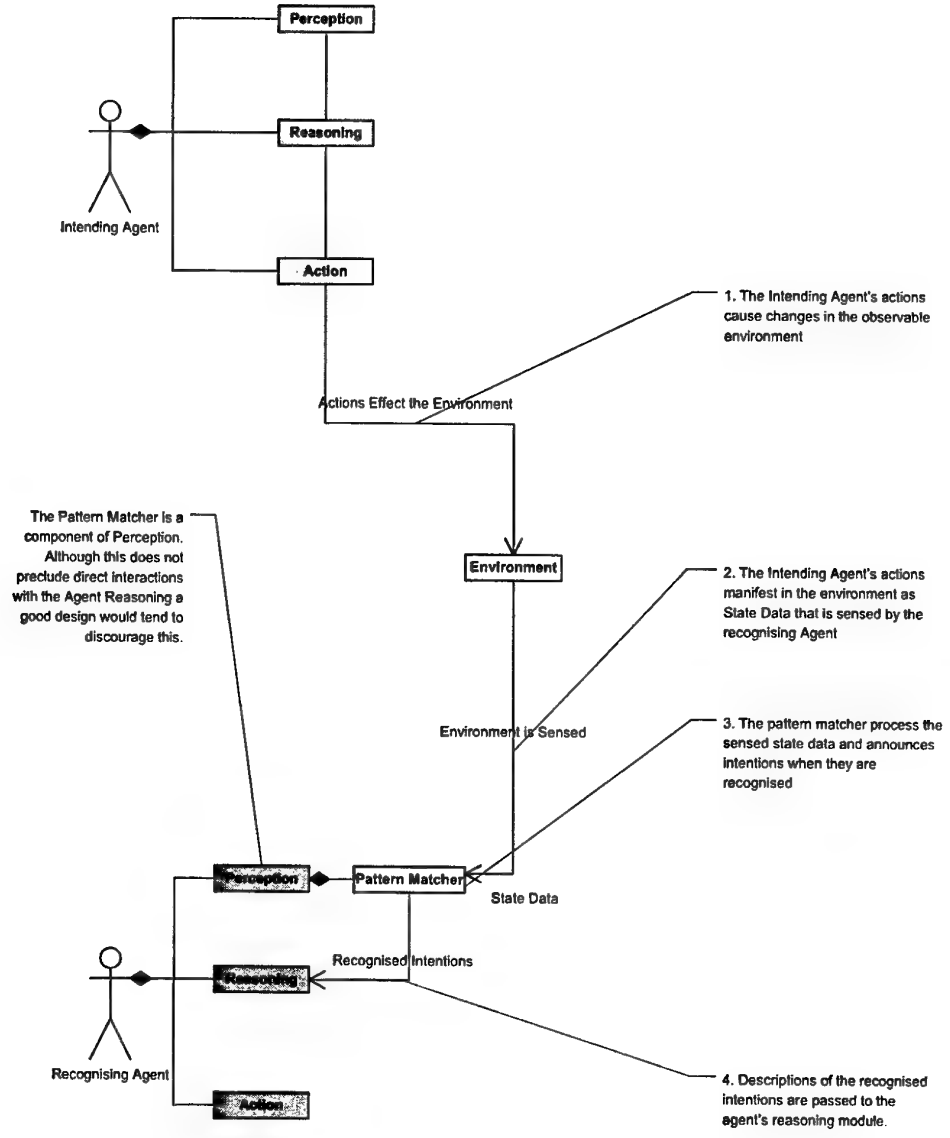

Figure 6.5: Pattern Structure: Ecological Recogniser.

announced as recognised intentions. The agent is substantially simpler in design and its reasoning component is simpler. The addition of pattern matching technology complicates the system.

6.2.4.7 Variants

There are a substantial number of variants possible with this pattern. Variants introduce feedback between the perception module and its pattern matcher and the reasoning module and its general reasoning. Two obvious variants are:

1. **Ecological Recogniser with Feedback** The pattern matching can be improved if the agent reasoning provides feedback to guide the process.
2. **Sub-Cognitive Ecological Recognition** The pattern matcher can be converted into a model of the sub-conscious aspects of the agent with the reasoning handling the cognitive aspects. This provides a variant of the pattern that provides a sophisticated cognitive model that can be used to model a variety of behaviours, including intention recognition (See Heinze et. al. [72]).

6.2.4.8 Consequences

The benefits of the Ecological Recogniser pattern are:

1. **Modularity** Intention recognition is a separate, self contained module within the agent's perception. This isolates intention recognition from agent reasoning and associates it with perception.
2. **Ontological purity** The agent is free to reason with the knowledge level concepts that are the output of the intention recognition process. The detailed information often required for intention recognition is encapsulated by the perception module and abstracted and processed by the pattern matcher. The agent receives and manipulates an appropriately high-level representation of the data.
3. **Fuzziness** Depending upon the implementation technology selected it may be easier to implement probabilistic it is easier to make the intention recognition process probabilistic.
4. **Technological Suitability** the separation of intention recognition from the rest of the agent allows the option of adopting a different technology for implementing intention recognition. In this sense it supports the adoption of the 'best tool for the job'.

and the liabilities:

1. **Modularity** although the advantages of modularity apply there is consequence that can, under certain circumstances, be significant. By separating the intention recognition process from the agent's reasoning it is more difficult to implement significant

interaction between the agent's practical reasoning and the intention recognition process. If the intention recognition process should be influenced by the agent's reasoning then this pattern is contra-indicated.

2. **Lack of Cognitive Plausibility** Although there is some limited support for this model of intention recognition in the psychology literature it is difficult to integrate this approach to intention recognition within a plausible cognitive model.

3. **Technological Complexity** Solutions that adopt the Ecological Recogniser pattern may choose dissimilar technologies for the various components—indeed the pattern encourages this. Although this may result in the selection of appropriate technologies there is a substantial increase in system complexity that results from agent constructed with hybrid technologies.

4. **Visibility** By encapsulating the intention recognition process some of the visibility into the intention recognition process is lost.

5. **Performance** Compared with the other intention recognition patterns, the Ecological Recogniser is likely to result in solutions that are computationally expensive.

6.2.4.9 See Also

The two closest relatives of the Ecological Recogniser are its simplification, the Assisted Ecological Recogniser, and its two most obvious alternatives: the Hybrid Recogniser and the Sense and Infer patterns.

6.2.5 Pattern: Assisted Ecological Recogniser

This pattern is a variant of the Ecological Recogniser pattern. The recognising agent is assisted in its task by virtue of direct access to named representations of actions in the environment. The Assisted Ecological agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent perceives directly the intention of the Intending Agent *in the patterns of activity of the Intending Agent. The example section is omitted for this section because there is no known example of it. In the following Chapter a description of a possible design for a system is introduced.*

6.2.5.1 Context

The provision of intention recognition in an intelligent agent system.

6.2.5.2 Problem

When access to the actions of the other agents is available and the relationship between the patterns of those actions and the intentions that cause them is complex then the

Assisted Ecological Recogniser provides a simplification to the Ecological Recogniser. It is a fundamentally different architecture.

The Assisted Ecological Recogniser pattern should be used under the following conditions:

- ontologies well defined, explicit and available
- action explicitly represented at run-time
- Only useful if there is a large number of actions and they are available
- Human using a GUI is a candidate for such a pattern. Button clicks/menu selection are the actions and are likely to be accessible rather than resort to the effects that those menu selects have. Agent has access to labels of human actions if the gui can provide a stream of the mouse clicks etc. One of the few cases where human actions are directly accessible to the agent. Accessing joystick/mouse/keyboard is less useful perhaps than the resulting state data in something like a computer game.

6.2.5.3 Solution

The assisted Ecological Recogniser is a modification of the Ecological Recogniser pattern that mirrors the modification of the Sense and Infer pattern to generate the Assisted Sense and Infer pattern. The basic concept of the solution is to ease the pattern matching intention recognition process by providing direct access to the action state.

6.2.5.4 Structure and Dynamics

The structure and the dynamics of the Assisted Ecological pattern are shown in Figure 6.6.

6.2.5.5 Variants

Like the Assisted Sense and Infer the variants of this pattern relate to the information that is provided to assist the Pattern Matcher. Rather than action data it might be useful to send plan data, function data etc.

6.2.5.6 Consequences

The Assisted Ecological Pattern has all of the benefits and liabilities of the Ecological Recogniser pattern with the following additions. The single primary benefit of the patterns is

1. **Computational Simplicity** the intention recognition process is simplified through the supply of a pre-recognised set of actions.

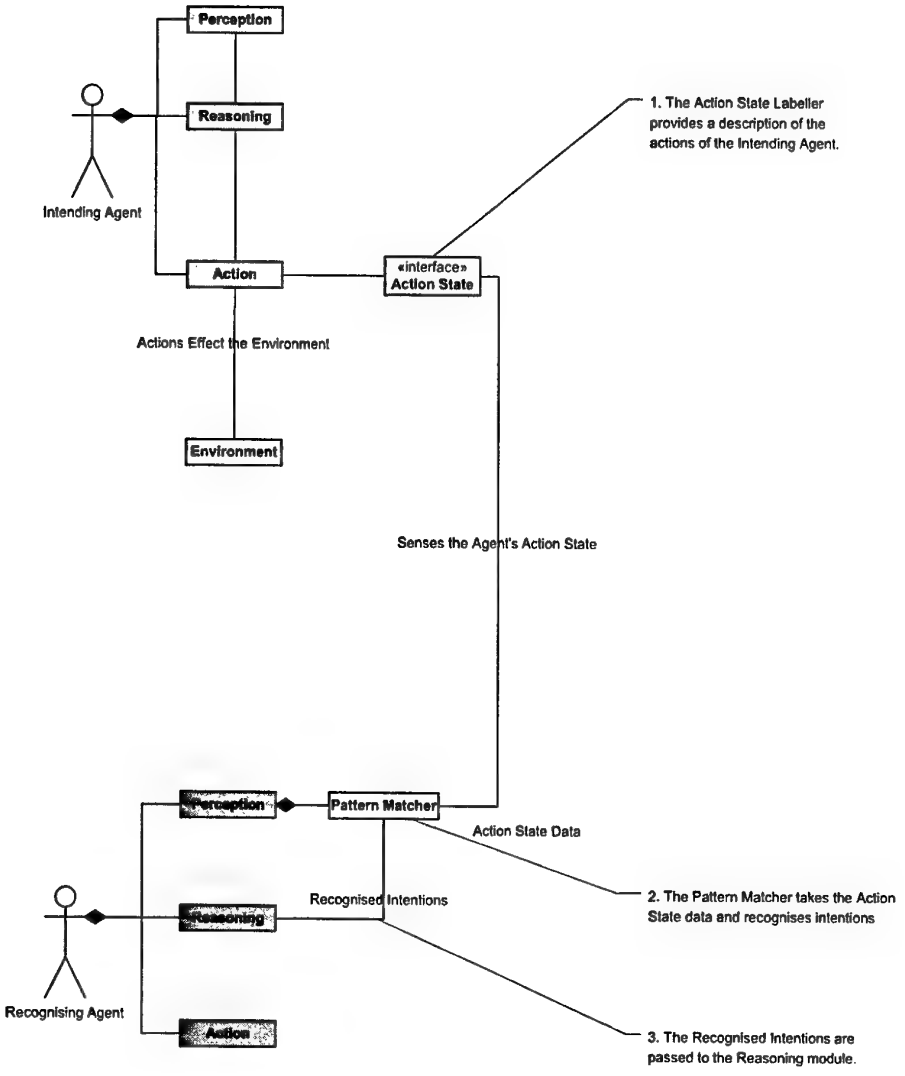

Figure 6.6: Pattern Structure: Assisted Ecological Recogniser

And the pattern also suffers from liabilities.

1. **Usefulness** Though the pattern is included here for completeness the practical use of this pattern will be limited to applications where the number of actions is large and also accessible. Few domains have these properties. Perhaps the best examples are those where humans interact with systems via a HCI.

2. **Doesn't make things easier** The Assisted Ecological pattern may not improve or ease the task of pattern matching. It might be that the detailed state data is actually more useful in disambiguating the intentions than the action data. Preliminary results from air combat simulation suggest that this is sometimes the case.

3. **Coupling** the Assisted Ecological pattern simplifies the processing of the pattern matcher at the expense of coupling between the agents. The Intending Agent and the recognising Agent are coupled and any modification to one of the agents will imply a requirement to modify the other.

6.2.5.7 See Also

The Assisted Ecological pattern is a simplification of the Ecological Recogniser and adopts a similar simplifying process to the Assisted Sense and Infer pattern.

6.2.6 Pattern: Clairvoyant

The clairvoyant pattern is named because the recognising agent simply 'becomes aware' of the intention of the other agents. It is neither sensed in the environment nor is it conceptually communication. It is at least superficially agent extra-sensory perception. The Clairvoyant agent architectural design pattern implements intention recognition in an agent system. The Recognising Agent directly senses the intention of the Intending Agent. This pattern is named because one agent simply 'becomes aware' of the others intention without any resort to deductive, inferential or other forms of reasoning or through any perceptual interaction with the environment. The awareness of the intention of the other agent is not ascriptive and nor is it sensory. Rather than a distinct pattern this design is a set of functionally similar patterns which are all variants on the same theme.

6.2.6.1 Example

During the development of SWARMM, the military simulator described in the introduction, it became apparent that modelling team behaviour was problematic [170]. In real life fighter aircraft on the same side when operating as a team normally remain within visual range. This allows them to coordinate behaviour without using the radio by observing each others movements. When one aircraft in a team manoeuvres it is usually immediately obvious to those other team members in the vicinity what the intention is. Simulating this behaviour with intelligent agents requires some form of intention recognition however

the implementation, if faithful to reality is extraordinarily complex. Furthermore, unlike modelling intention recognition of opponents where ambiguities might be a result of deliberate deception or evasion and in real life there would be much uncertainty, in real life pilots are less likely to misinterpret their team members actions. This renders the high fidelity simulation of intention recognition of team members largely a waste of resources.

6.2.6.2 Context

The provision of intention recognition in an intelligent agent system.

6.2.6.3 Problem

When design simplicity and operational performance are high priorities and there is scope for the system outside of the Recognising Agent to be designed, it is possible to implement intention recognition by adding a *labeller* to the agents that are to be recognised. This labeller will provide an appropriate, abstract and simple representation of the intention of the agent in a form that is suitable for the Recognising Agent to access.

There are cases where the intention to be recognised is objective and will be the same for all agents that are attempting recognition. In these cases there is little need to duplicate the intention recognition functionality in all of the recognising agents when it can be placed into the agent that has the intention. Even if the intention is recognised in different ways by different agents this can be catered for with a slightly more sophisticated labeller.

The Clairvoyant pattern should be used under the following conditions:

- The system, apart from the Recognising Agent, is able to be modified. Specifically it is possible to modify the system to provide the Recognising Agent with direct access to a representation of the intention of the other agents.
- The recognised intentions do not depend upon the mental state of the Recognising Agent.
- The specific details of the intention are not required—a high level description is adequate.
- Design simplicity and system performance issues will dominate.

6.2.6.4 Solution

By adopting the Clairvoyant pattern the design provides the Recognising Agent with a direct representation of the intention of the Intending Agent. The Clairvoyant pattern employs a *labeller* to provide direct access into the intentions of the Intending Agent. Conceptually the Recognising Agent simply reads a label that describes the intention of the agent but the detail of the implementation can vary widely and depends upon the particular internal structure of the intending agent. The possibilities are discussed below under *Variants.*

6.2.6.5 Structure and Dynamics

The structure and dynamics of the Clairvoyant pattern are shown in Figure 6.7.

6.2.6.6 Example Resolved

In reality there are many cases where there will be little or no ambiguity in recognising the intention of a team member and for these cases it was determined that modelling intention recognition by granting one agent direct access into the into internal state of another was legitimate. When one agent adopted an intention it was broadcast via a dedicated communication mechanism to all other members of the team that were currently within visual range. This was different to the normal channels of communication used to simulate radio transmissions and didn't make use of the standard ontology of communication. It was a design 'shortcut' to eliminate the need for sophisticated modelling of intention recognition in a case where it was deemed to be unnecessary.

6.2.6.7 Variants

Variants of the Clairvoyant Pattern relate to the manner in which the labeller is implemented. Although the pattern can be conceptualised as a labeller the variants are architecturally quite different. These variants correspond to the different forms that the 'labeller' might take as described in Section 6.2.

1. **External Labeller** The Recognising Agent is supplied with a representation of the intention of the intending agent by a process external to either agent. The external labeller acts as a third party model of all of the intention recognition functionality required by the system.

2. **Agent Specific Labeller** In this case the labeller (rather than simply supply a representation of the intention of the intending agent) takes into account the identity of the Recognising Agent and supplies a specific label.

3. **Communication** Rather than 'label' the agent the Intending Agent communicates its intention to the Recognising Agent. Though not really intention recognition in a strict sense it is credible possibility as a *model of* intention recognition.

4. **Interrogation of Intent** Similar to the 'communication' case above but the two agents enter into a dialog about the nature of intent. This offers the possibility of introducing subjective intention recognition.

6.2.6.8 Consequences

Some of the benefits of the Labelled Intention Recognition Pattern are:

1. **Simplicity** As the simplest imaginable implementation of intention recognition it is almost the 'null case' and is arguably better described as intention notification.

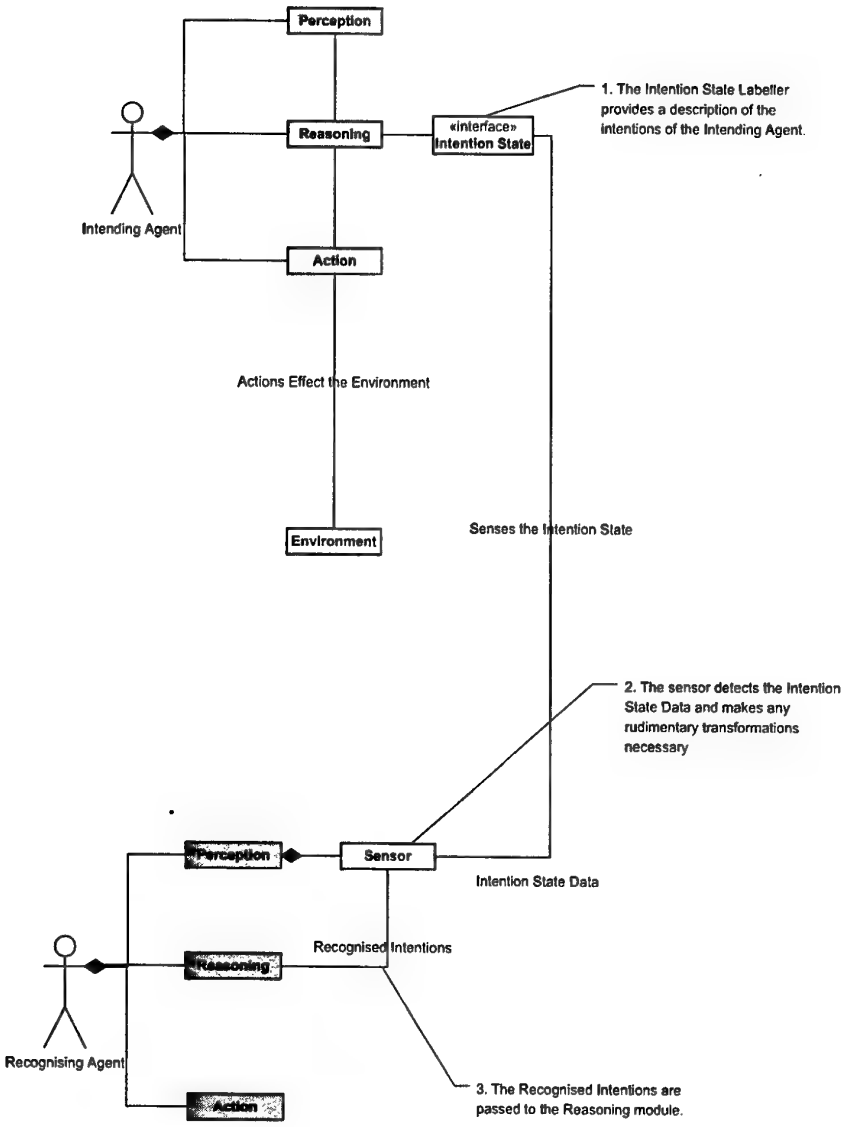

Figure 6.7: Pattern Structure: Clairvoyant

None the less it still offers the prospect of modelling intention recognition when simple system and agent architectures are required.

2. **Performance** As a result of the simplicity of the architecture and the complete short circuiting of the intention execution and intention recognition processes present in the other patterns the Clairvoyant will almost certainly offer the fastest performance of any of the patterns.

3. **Visibility** The straight-forward nature of the Clairvoyant pattern means that although it might lack *realism* it is simple to understand, debug, and explain. The functioning of intention recognition under this pattern is clearly visible. In systems that require well validated, well understood, and robust behaviour this is a distinct advantage.

4. **Ontological Purity** The Recognising Agent is free to reason with only high level representations intention. There is no requirement that the agent manipulate low-level data such as the environment state.

and the liabilities:

1. **Loss of Autonomy** Intention recognition is now (at least partly) modelled outside of the agent undertaking the recognition

2. **Coupling** Agents are no coupled with other agents. This coupling is more or less strong depending on the particular variant

3. **Cheating** Little cognitive plausibility. It is a model of intention recognition but is it really intention recognition.

4. **Lack of Flexibility** A change to any one agent may well require changes to all others.

5. **Designability** The Clairvoyant pattern requires that aspects of the system outside of the Recognising Agent be *designable*.

6.2.6.9 See Also

The Clairvoyant is the logical extension of the simplifications of the Assisted Sense and Infer and Assisted Ecological patterns.

6.3 Summary

This Chapter has presented a catalog of six patterns that collectively provide a range of solutions to implementing intention recognition. Experience with these patterns is limited but the presentation in this form has several benefits. The *look and feel* of the object-oriented patterns literature was adopted and additionally the UML was used to present

the structural descriptions of the patterns. These patterns are then used as the basis for the actual systems described in detail in Chapter 7.

The criteria introduced in Section 6.1.3 can now be used to evaluate the relative advantages and deficits of each of the patterns. Clearly there is substantial variability in the patterns possible when the detail is established as the system is constructed. This evaluation should not be considered to be rigid, a pattern described as *computationally simple* might in certain cases result in a system that is actually quite complex, rather the evaluation provides descriptions of the types of system that the pattern is *likely* to support, or the types of systems likely to result from application of these patterns. Figure 6.8 summarises the evaluation criteria discussed below.

Module Complexity Each of the patterns will lend themselves to designs and implementations that result in more or less complex solutions. Considering just the *perception* and *reasoning* modules of the agent provides an interesting comparison of the agents. Clearly the Clairvoyant pattern is the simplest with an almost trivial perception and reasoning module. The Ecological pattern has similar reasoning module to the Clairvoyant (i.e. trivial) but places all of the functionality of intention recognition onto the perception module. Perception complexity is removed by the *Assisted Ecological* pattern which can be seen as an intermediate between the *Clairvoyant* and the *Ecological* patterns. The *Sense and Infer* pattern reverses the *Ecological* pattern by placing the intention recognition functionality into the agent reasoning resulting in a very perception module at the expense of reasoning complexity. The *Assisted Sense and Infer* pattern reduces the complexity of the perception module without complicating the reasoning module. Finally, the *Hybrid* pattern has a similar perception module to the *Assisted Sense and Infer* pattern with some added complexity in the perception module.

Module Cohesion Module cohesion is an excellent criteria for assessing an architecture. Software engineering tells us that for reuse, maintenance, testing and many other reasons high cohesion is desirable [176]. There are two types of cohesion that might be considered here. One is the *agent system cohesion*—the degree to which intention recognition is internal to an agent or spread throughout the system. The second is the cohesion of the modules that are internal to the agent—and hence a measure of the extent to which the intention recognition functionality is spread throughout the modules internal to the agent. Three patterns, *Sense and Infer*, *Hybrid*, and *Ecological*, isolate the intention recognition functionality inside the agent. The *Sense and Infer* pattern has the added advantage that the decomposition of functionality internal to the agent separates the perception and inference aspects of intention recognition in a manner that the *Hybrid* and *Ecological* patterns don't quite achieve. The other three patterns spread the intention recognition functionality across the agent system and so score poorly.

Agent to Agent Coupling Low coupling is the software engineering corollary of high cohesion and is desirable for many of the same reasons [176]. Three patterns, *Sense and Infer*, *Hybrid*, and *Ecological* have no agent-agent coupling related to the provision of intention recognition. In each case interactions occur via the environment. The two *Assisted* patterns require the sharing of information between agents with

reference to the activities of those agents and are coupled at the level of agent activity. The *Clairvoyant* pattern is coupled at the intentional description level which ties the functional implementations of the agent even more tightly together.

Implementation Complexity Estimating the complexity of an implemented system based on these patterns is difficult but from with some experience of their application and informed guesses based on the likely implementation technologies required for a non-trivial application it is likely that the Clairvoyant system will be the simplest, the two assisted patterns will follow and the other three being all relatively more complex but impossible in general to distinguish from each other. In the following section examples of designed and implemented systems illustrate this point more clearly and describe examples of these implementation technologies.

Data Provided to the Perception Module The perception module is "skinny" for three of these patterns, little more than a conduit for data entering the agent. The data that is presented to the perception module (and hence the agent) is the environment state for the *Hybrid*, *Ecological*, and the *Sense and Infer* patterns. The two *Assisted* patterns are presented with the agent state. The *Clairvoyant* pattern receives the agent intention state directly.

Data Provided to the Reasoning Module Three of these patterns, Clairvoyant, Ecological, and Assisted Ecological have no component of agent reasoning devoted to intention recognition. In those cases intention is determined prior (either externally to the agent or in the perception module) and provided directly to the reasoning model. The *Sense and Infer* pattern reasons directly with environmental state data. The *Hybrid* pattern and the *Assisted Sense and Infer* pattern both reason with action state data.

Data Processing The data processing criteria examines the amount of data transformation required to arrive at a recognised intention. Those patterns that maintain high-level data descriptions explicitly in the system to support intention recognition will clearly do better against this metric. The *Clairvoyant* pattern requires almost no data processing. The two *assisted* patterns have systems that provide explicit access to action state data and require a little more data processing than the Clairvoyant pattern but less than the remaining three.

Computational Performance Estimating the implemented computational performance of these patterns in anything but the most general terms is impossible. The *Clairvoyant* is likely to be best, followed by the *assisted* patterns. This is similar to and not unsurprisingly closely related to the Implementation complexity criteria discussed above.

Architectural Complexity The architectural complexity criterion combines the coupling and cohesion criteria and also considers the nature of the information exchange necessary between the modules of an implemented system. The Hybrid pattern scores highest here because of the comparatively large amount of information exchange likely to occur between the modules. The *Clairvoyant* pattern is clearly the simplest. The other patterns are more difficult to distinguish.

Amount of Processing in Reasoning One of the secondary aims of this thesis was to explore agent designs that reduced the complexity of the agent by moving aspects of the design into the environment. One criteria by which the success of this might be judged is to examine the amount of processing related to intention recognition that is conducted by the agent reasoning module. *Clairvoyant*, *Ecological*, and *Assisted Ecological* score well against this criteria with less processing performed inside the agent's reasoning module. The *Assisted Sense and Infer* and the *Hybrid* follow, with the *Sense and Infer* pattern, which does all of the intention recognition processing in reasoning last.

Cognitive Model Though not an aim of this thesis each of the patterns support, at least partially, some psychological theory of cognition. For example, the *Ecological* pattern supports directly the basic ideas of ecological psychology, albeit in an extreme form. The *Sense and Infer* pattern corresponds to a more mainstream view of psychology that regards perception as very low level. The *Hybrid* pattern holds a reasonable middle ground between these two areas of psychology and accords most closely with the ecological psychologists view of cognition.

Location of Intention Recognition Considered as a single functionality it is intuitive to expect that intention recognition would be located, in an architectural sense, in a single place, resulting in a system exhibiting high cohesion. A major thrust of this thesis, and as a result, several of the patterns, is that the intention recognition functionality can be spread across the agent system. Figure 6.8 indicates the location of the intention recognition for each of the patterns but of primary interest is the idea that for three of the patterns, *Clairvoyant* and the two *Assisted* patterns, at least part of the function of intention recognition is located outside of the agent.

The patterns are closely related variants of an architectural design theme that arose from a consideration of the environment as being a designed part of the agent system. Figure 6.9 shows the relationships that exist between these patterns. Pattern maps provide a means of improving understanding of software patterns by describing the relationships that exist between them [18].

Figure 6.10 is provided as a guide to pattern selection but the detail of the particular requirements of any given system will almost certainly dictate a more careful, detailed examination. Recent research in this area [117] is adopting a means of developing assessments of the designers preferences in choosing patterns. This type of approach is a more sophisticated evaluation of the simple, arbitrary, and pragmatically grounded guides shown here.

Pattern selection guides such as that shown in Figure 6.10 are likely to be a significant part of agent patterns as they provide an indexing into the agent patterns necessary to cater for the greater number of closely related patterns that will occur in agent systems development. The cognitive patterns literature [55] is foreshadowing this trend.

		Hybrid	Sense and Infer	Assisted Sense and Infer	Ecological	Assisted Ecological	Clairvoyant
Module Criteria	Module Complexity	Perception Medium Reasoning Medium	Perception Low Reasoning High	Perception Low Reasoning Medium	Perception High Reasoning Low	Perception Medium Reasoning Low	Perception Low Reasoning Low
	Functionality/module match	Good	Poor	Medium	Good	Medium	Poor
	Cohesion	Medium	Low	High	Medium	Medium	Medium
	Agent to Agent Coupling	Low	Low	Medium	Low	Medium	High
	Implementation Complexity	High	High	Medium	High	Medium	Low
	Implications for System	Hybrid technologies	Everything is in the agent reasoning	Action state must be accessible	Pattern matching technology required	Action state must be accessible	Intention state must be accessible
Data Criteria	Data Provided to Perception	Environment state	Environment state	Agent action state	Environment state	Agent action state data	Agent intention data
	Data Provided to Reasoning	Agent action state	Environment state	Agent action state	Agent intention	Agent intention data	Agent intention data
	Data Processing Required	High	High	Medium	High	Medium	Low
Process Criteria	Likely Computational Performance	Low	Low	Medium	Low	Medium	High
	Architectural Complexity	High	Medium	Medium	Medium	Medium	Low
	Total system complexity	High	Medium	Medium	High	Medium	Low
Cognitive Modelling Criteria	Amount of processing performed in the agent reasoning	Medium	High	Medium	Low	Low	Low
	Cognitive Model	Actions are perceived directly and then intentions is inferred	Actions and intentions are both reasoned about	Actions are immediately accessible and then intentions is inferred	Intentions are directly perceived	Actions are directly accessible and then intention is inferred	Intention recognition is telepathic
	Location of Intention Recognition	Perception and Cognition	Cognition	Other agent and Cognition	Perception	Other agent and Perception	Other agent

Figure 6.8: Detailed Comparison of Pattern Attributes

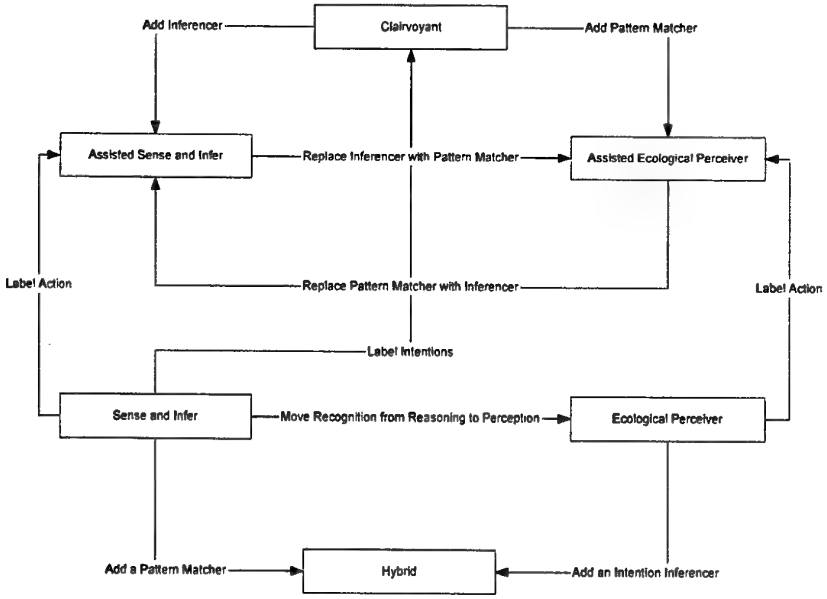

Figure 6.9: A pattern relationship diagram is one way of visualising the relationships between a set of patterns. One disadvantage of this type of diagram is that there is no clear starting point. This is addressed by pattern selection diagrams of the type shown in Figure 6.10.

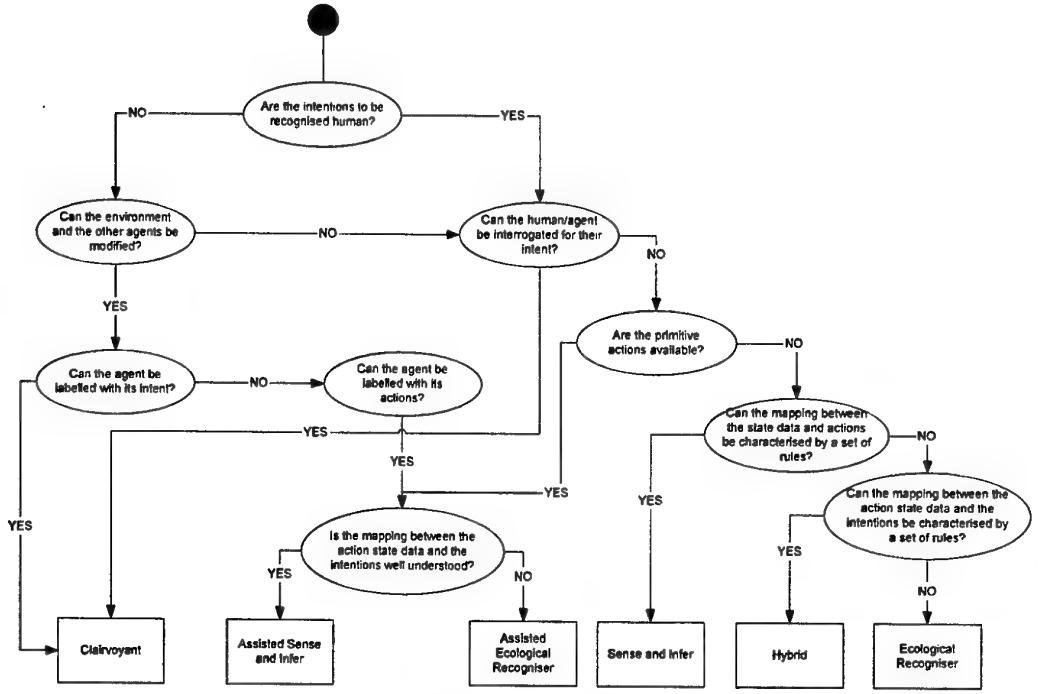

Figure 6.10: Selecting the appropriate design pattern can be assisted by traversing the decision tree.

Chapter 7

The Virtual Air Show

"Learning to fly does not take long—within the first 20 hours of flying you will have learned the basic skills. [...] Patterns formed in the first few hours will stay with you throughout your flying life."—Trevor Thom [167]

By constructing a number of related but functionally different agent systems it is possible to demonstrate the engineering utility of the themes discussed in Part 2.8 and to gain experience in the application of the patterns of Chapter 6. This chapter illustrates the methodology in practice through the architectural design of a family of flight simulators, referred to collectively as the Virtual Air Show (VAS). Six example architectures are described, corresponding to the patterns that were presented in Chapter 6.

Section 7.1 very briefly introduces flight simulators and summarises the characteristics of the problem domain that are general to the six architectural designs that follow in Section 7.2.

Section 7.2 describes the architectures of the six variants that result from the application of the design patterns. This description is only detailed enough to illustrate the application of the pattern and highlight the strengths and weakness.

Two of the patterns *Sense and Infer* and *Ecological Recogniser* are the most technically challenging to construct. From an implementation perspective the other patterns are simplifications, hybridisations, or modifications of these two patterns. Examples of the *Sense and Infer* pattern are found in the literature and have been described elsewhere in this thesis (Section 2.3). Design descriptions are adequate to compare and contrast the application of the six design patterns but a complete implementation of the Ecological Recogniser is presented in Section 7.3 for three important reasons:

1. The elaboration and the weight given to models of perception by this thesis was based upon the assumption that 'direct perception' of intention (or some abstract properties of agent systems) was possible. Certainly it is theoretically possible but in a thesis with a software engineering focus *theoretically possible* is insufficient. It is necessary to show that the architectural benefits claimed for this pattern are realisable in a practical system. If no technology exists to actually implement the pattern it is of no practical use and the credibility of all related the design patterns could be questioned.

2. It is technically challenging and, if implemented successfully, provides evidence that the other related and simpler patterns could also be implemented.

3. Some of the implications of design choices are only available by an inspection of a completed application.

Section 7.2.7 discusses the results of this chapter as they pertain to modelling intention recognition in a broader context.

7.1 Introduction

Flight simulation was chosen as the domain for the demonstration of the patterns because it embodies many of the requirements that drive the modelling of intention recognition in intelligent agent systems. A primary consideration was the obvious similarity to the military simulations that fostered this thesis and the ease with which the lessons learned could be mapped back into the military context. Even without this strong organisational consideration there are sound technical reasons for the choice. Plausible examples of the application of the patterns are applied to model intention recognition of both humans and agents. Flight simulation provides a rich environment that must be perceived and generates a substantially complex intention recognition task simply by the sheer weight of information available for processing. Flight simulators almost always operate as real-time or faster than real-time systems placing strong performance requirements on the provision of acceptable designs. Performance considered in the context of the available technology can play a strong role in the design of systems and so an example that pushes the practical performance limits is to be preferred. Further details of these systems have been published elsewhere [71, 72].

The basic components of flight simulators and the activities associated with circuit flight in this section help in setting the context for the description of the architectures that follow in Section 7.2 and in providing the required detail for the more detailed elaboration of the Ecological Recogniser pattern in Section 7.3.

7.1.1 Flight Simulators

The VAS is a flight simulator [22]. It was proposed, designed, and developed as a test environment to explore issues related to perception and recognition in agent systems[55]. There is a strong relationship to military flight simulators such as those described in Section 6.1.4 though the VAS does not incorporate any modelling or representation of weapons or other military-specific aircraft systems.

Flight simulators come in all shapes and sizes. The highly sophisticated simulators operated by large civil airlines and air forces require significant budgets to acquire and operate (See Fig. 7.2). These simulators are designed for training pilots in procedures that are too dangerous or too costly to perform in real aircraft. Simpler and cheaper

[55]Some of the development of the simulation kernel, the environment, and the graphical user interfaces was undertaken by Adrian Pearce at Curtain University.

HOTASTA (hands on throttle and stick training aids) are available to provide training in basic procedures of operating the aircraft without the need for the complex modelling of the complete aircraft—these are often referred to as *switchology trainers*. Cheaper and simpler still are the simulators that populate the computer games market [31]. Literally hundreds or these are available with varying level of fidelity and functionality. Flight simulators can be put to many purposes. The most familiar application outside of computer games is for the training of airline pilots in emergency procedures that cannot otherwise be trained for. Similarly military pilots utilise them for rehearsing missions, training procedures, or developing skills. Flight simulators share a number of basic components:

User Interface The most visible of the these is the UI that provides the pilot with some representation of the cockpit and the environment. Sophisticated flight simulators may have physical mock-ups of the actual cockpit to help create an immersive experience for the pilot whilst many may provide a virtual representation via a more GUI on a single computer screen. The latest flight simulators include virtual reality helmets and gloves and even suits that inflate to simulate the presence of g-forces. The VAS utilises a single computer screen displaying the cockpit instruments and a simple 3d out of the cockpit view that includes some representation of trees, terrain, buildings, and a runway.

Aerodynamic Model The aerodynamic model provides the flight and handling characteristics of the aircraft. This might include force feedback into the controls and realistic performance and handling characteristics over a wide range of flight conditions. The aircraft aerodynamics can be simulated to varying levels of fidelity. A detailed description of the aerodynamic techniques is beyond the scope of this thesis but in order to provide a realistic representation of aircraft behaviour for the purposes of testing recognition a medium to high fidelity model was chosen. The implemented VAS system described in Section 7.3 uses a sophisticated high fidelity aerodynamic model sourced from the RAAF's PC-9 training simulator 7.1. The aerodynamic model provides data to the simulation environment about the current state of the aircraft. This includes the position, velocity, angular and linear rates, and the control positions. This is packaged for the agents in a form that reflects the information available to a real pilot in a real aircraft by observing the flight instruments or by looking outside.

Environment The model of the environment provides the weather, atmosphere, the buildings and runways, terrain, and the landscape. This can vary from the sophisticated 3-D worlds present in both computer games and military simulators to simple vector graphics representations for specific purposes. The VAS uses a simple 3d world populated by terrain, simple buildings, roads, and an airport with a runaway. The terrain/buildings are specified via a file. This allows the location, orientation, and the very existence of all of the environmental entities to be rapidly modified.

Simulation Engine/Kernel The part of the software system responsible for scheduling the processing and polling of the other components. This might include interfaces with other systems. The VAS uses a time-stepped simulation engine that schedules

and executes each of the elements in turn. It is not as sophisticated as the event-based variable time-step kernels available but meets the requirements of this project.

In addition the Virtual Air Show incorporates two agents that fill the roles of flying instructor and pilot.

The Instructor Agent The Instructor Agent is an intelligent entity that fills the role of a flying instructor. It observes the pilot (whether human or agent) and provides advice about appropriate actions to take. If the instructor agent is to provide timely useful advice to the pilot then it is advantageous that the advice is provided in the context of the intended activities of the pilot. It is not simply enough to recognise the pilots actions and provide advice after the fact. In the context of the VAS the pilot's intentions are those associated with circuit flight and cover the types of activities mentioned in Sections 7.1.2 and 7.1.3.

The Pilot Agent The pilot agent is designed with a separation between the low-level activities of controlling the stick and throttle and the higher level *cognitive* activities of decision-making. From a cognitive modelling standpoint this is a natural split. In practical terms it is simpler to implement agent decision making in languages like dMARS, JACK, SOAR but these are less well suited to the fine control and mathematical processing often required when providing agent with the ability to control the low-level manoeuvring of an aircraft.

7.1.2 Circuit Flight

Every airport has a procedure for take-off and landing that defines a series of pathways in the sky that provide the routes by which the pilots leave and approach the airport (See Fig. 7.3). These pathways are standard across all airports anywhere in the world and allow for pilots to land at unfamiliar airports by following the standard operating procedures. These pathways are known as the *circuit* and a pilot is said to be *joining the circuit* when the aircraft flies onto one of these paths with the intention to land. A set of procedures for joining the circuit in order to land or leaving the circuit after take-off is a significantly important part of every novice pilots basic training [167]. During pilot training it is common for pilots to practice circuit flying by repeatedly taking-off, flying the circuit and landing without ever leaving the circuit or stopping the aircraft. The VAS was designed as a methodology demonstrator and makes use of the flight training domain to provide useful examples of the engineering practices proposed in this thesis.

Circuit flying involves taking-off, flying a roughly rectangular course and landing, see Figure 7.3. Some detail of the elements of a circuit are presented here as they provide valuable background for understanding the design choices presented in the following sections.

The basic components are:

Take-off and Climb The aircraft is powered up and flown into the wind along the runway. At an appropriate speed the pilot pulls back on the stick to raise the nose

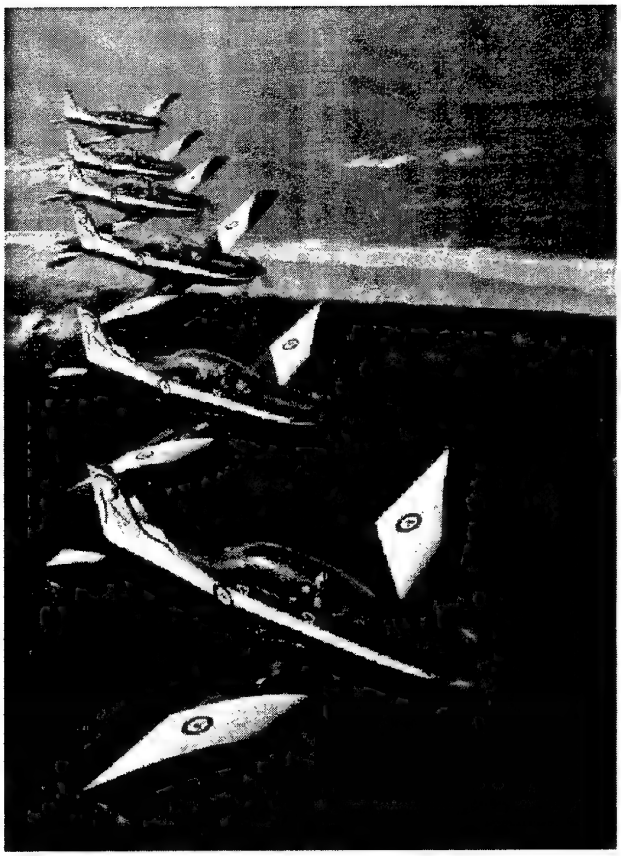

Figure 7.1: The aerodynamic model from a current PC-9 military simulator was used for the development of the VAS. The PC-9 is a current Royal Australian Air Force training aircraft. Shown here are the Roulettes, the RAAF's aerobatic display team that currently fly the PC-9 aircraft. The aerodynamic model provides a six-degree-of-freedom simulation of the aircraft and includes landing gear effects, flaps, and the dynamics of stalled flight. These features provide the detail necessary for simulating circuit flight.

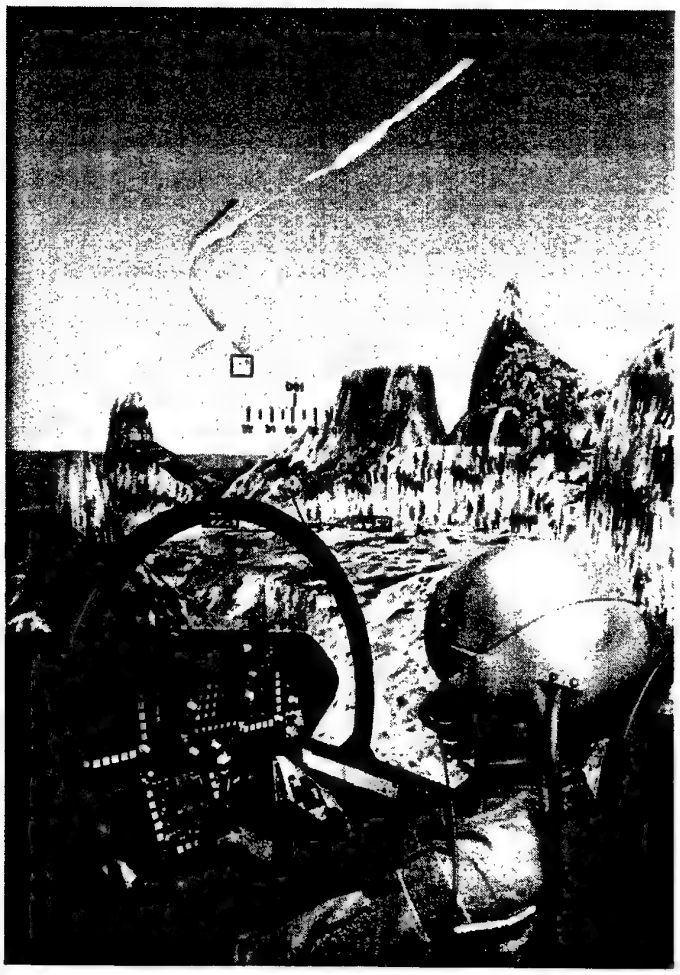

Figure 7.2: Military flight simulator typical of those utilised by modern defence organisations. These simulators provide comparatively safe and cheap training in procedures, tactics, and missions for combat pilots. An increasingly important aspect of these simulators is the provision of computer generated forces as virtual opponents. Intention recognition has been identified as an important addition to these systems.

of the aircraft and initiate the *lift-off*. Following this is the *initial climb* to a safe manoeuvring height. During this climb the landing gear will be retracted and the flaps are returned to their undeployed setting.

Turn to Crosswind Leg At a predetermined height, generally 500 feet above ground level (AGL) the aircraft is turned ninety degrees to the left[56]. This turn is a gentle turn, typically twenty degrees angle of bank. After this turn is complete the fuel pump, trim and flaps are checked and the climb is continued until circuit height is reached, usually 1000 feet AGL.

Crosswind Leg The crosswind leg requires the pilot to fly at constant altitude at right angles to the runway. Because the wind is blowing from the right hand side of the aircraft it may be necessary to slightly angle the aircraft into the wind to remain on course.

Turn to Downwind Leg The turn to *downwind leg* is a medium turn at constant circuit height (1000 ft AGL).

Downwind Leg The downwind leg is flown at constant altitude and speed. During the downwind leg the pre-landing checks are initiated and a radio call is made announcing the intention to land.

Turn to Base Leg Once the runway threshold passes abeam of the aircraft the descent may be commenced. Though it is more common to commence descent only after the turn to *base leg* is completed. One of the more difficult elements of the circuit is judging the point at which to commence the turn onto the base leg. The turn should be commenced when the touchdown point on the runway lies about 30 degrees behind the aircraft. In strong winds the turn should be commenced sooner as there will be a tendency for the wind to push the aircraft away from the runway as the turn is performed.

Base Leg and Turn to Final During the base leg a descent is commenced from 1000ft AGL to about 500-600 ft AGL. At an appropriate point a gentle turn onto final is commenced. At the completion of this turn the aircraft should be aligned with the runway and at a minimum of 500ft AGL.

Final, Descent and Landing The final should be flown at a constant rate of descent judged to cause the aircraft to touchdown near the beginning of the runway. The rate of descent is controlled with the throttle. As the aircraft approaches the touchdown power can be cut and the aircraft is 'flared'. The pilot pulls back on the stick so that the descent rate is reduced and the aircraft flies just above the ground, gradually losing speed until it touches down.

As part of the general documentation of the system domain the UML was used to document this knowledge of the domain and to highlight the aspects of the domain relevant to the flight simulation development. When simulation software is developed (or any software that models reality) it is important to maintain adequate documentation of the

[56]Left hand circuits (such as that shown in Fig. 7.3) are most commonly used at airports. Occasionally, when geography, traffic, weather or other conditions dictate, circuits may be flown to the right.

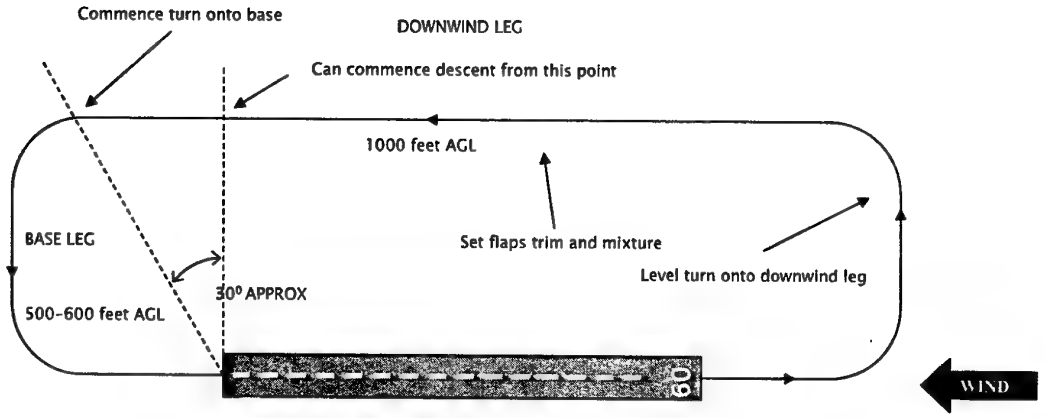

Figure 7.3: The basic elements of a standard circuit. The circuit can be joined at any point.

real-world aspects so that simplifying assumptions and modelling choices can be clearly documented[57].

7.1.3 The Circuit in More Detail—Turn to Base Leg

One of the more difficult judgements that a novice pilot must make when flying a circuit is the point at which to commence the turn onto base leg.

When to turn from the down wind leg onto the base leg is one of the more difficult decisions that a student pilot must learn to make. As this particular phase of the circuit will form the basis for examples throughout this chapter more detail is provided here.

The Downwind leg is a critical part of the circuit. It is flown parallel to the runway and allows the pilot time to commence the checks for landing, observe the condition of the runway and the wind gauge the wind strength and direction. During the downwind leg flaps trim and mixture are set flor the descent to land. Once the runway threshold passes abeam of the aircraft (as shown in Fig. 7.4) the pilot may commence a descent. When the runway is about 30 degrees to stern a medium turn onto base leg is commenced.

Difficulties arise in becoming familiar with the geometry and judging the turn point because subtle shifts in wind direction can vary the observed angles as the aircraft. Even if the wind is from directly astern the aircraft it is important to accurately judge the wind strength. If the wind is strong it will push the aircraft further down range of the runway during the turn. So that the aircraft remains close enough to the runway the turn should be commenced earlier.

Though there is a natural predisposition to use landmarks at familiar airports to judge the turn points pilots are trained to estimate wind strength and direction and use only the runway to judge the best turn-point.

[57]It is left for future research to develop the case for the UML as a modelling language for capturing simplifying assumptions in simulation development.

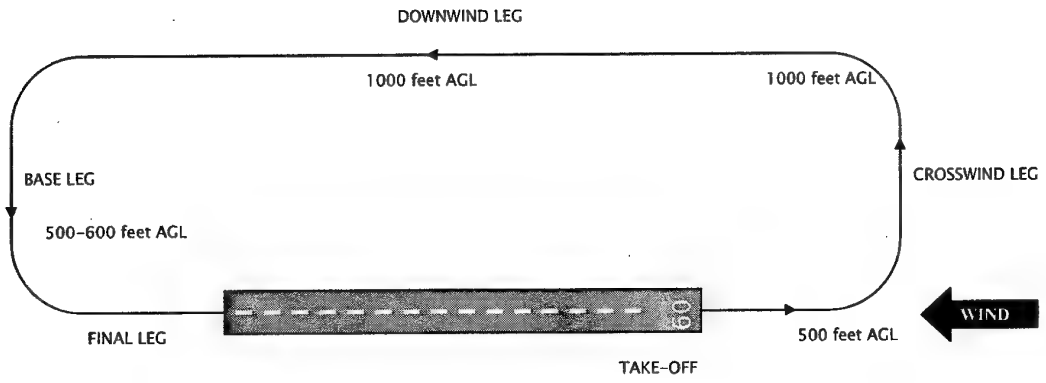

Figure 7.4: The important components of the down wind leg of the circuit.

7.1.4 Intention Recognition in the Virtual Air Show

Intelligent agents are incorporated into the Virtual Air Show in two roles: as flying instructors; and as pilots. Variants of the system that allow humans to fly the aircraft are included. The core functionality associated with instructing the pilot and reasoning about the appropriate advice to give is written in some desirably high-level agent language using high-level agent concepts. Intention recognition is provided to the Instructor Agent to improve its ability offer timely advice to the pilot. The intentions that the instructor must recognise are those related to the behaviour of the pilot and the actions that are taken whilst flying the aircraft around a circuit. The full complexity of the possible intentions are not addressed in this thesis but the description of the circuit flight in Section 7.1.3 provides an indicative example of the types of behaviour associated with intention recognition.

7.1.5 Virtual Air Show Variants

The flight simulation described here is just one of the many types of agent systems that might require intention recognition. The obvious application domains are; computer games; military simulation and threat assessment systems; and personal digital assistants. The implications of this research for these domains are discussed in Chapter 8.

The distinguishing feature of the VAS is the provision of an agent within the system responsible for acting as a flying instructor for the pilot. There is evidence that an intention recognition capability supporting instruction of this kind greatly improves their ability to provide timely relevant advice. Whether or not this claim is justified is outside the scope of the thesis, but assuming that there is some basis for it, a requirement of the VAS is that the instructor agent have the ability to recognise the intention of the pilot. The following examples then illustrate different variants of the VAS that both characterise the types of constraints that might be placed on agent systems and the engineering solutions to intention recognition that might result when the six patterns developed earlier in the thesis are applied.

To illustrate the simplifications that can be made when engineering agent based systems

that offer more design freedom the second, fourth and sixth builds move the role of the pilot from that of a human interacting with the system to that of an intelligent agent operating inside the system. With the 'pilot' now an agent within the system there is potential to *short circuit* the intention recognition process by direct inspection of the pilot's internal states.

7.1.5.1 Version One: Ecological Recogniser

In the first example (Section 6.2.4) an interactive system is presented[58]. The functionality of the first build of the VAS is not that dissimilar from commercially available flight simulators for personal computers. The notable difference is in the addition of a software agent to guide the pilot through the flight. This agent will be referred to as the 'instructor-agent' and plays a role similar to the instructor of a student pilot learning to fly circuits. Amongst other functionality the instructor-agent must recognise the intention of the human pilot so as to best assist in the activities associated with taking off and landing a light aircraft. It is the description of the design and implementation of this intention recognition that is the focus of this Chapter. The agent is watching the pilot's manoeuvring and determining the best advice to give based upon the recognised intentions. The agent must recognise the pilot's intention as early as possible, so that the advice might be timely. The agent does not know in advance what the pilot will do and there is no communication from the pilot to the agent. Messages from the agent to the pilot are conveyed via a speech synthesis utility. This example illustrates the application of the Ecological Recogniser pattern from Section 6.2.4.

7.1.5.2 Version Two: Assisted Ecological Recogniser

The Assisted Ecological Recogniser pattern from Section 6.2.5 results in a design which is closely related to the first variant. Two variants of this pattern are described. One in which the pilot is an intelligent agent and one in which the pilot is a human. These two examples illustrate the possibility for the recognition process to gain access (in some cases) to a representation of human action.

7.1.5.3 Version Three: Sense and Infer

The third example redesigns the VAS to integrate a reactive intention recognition system into the agent reasoning. This has advantages. It allows the cognitive aspects of the agent's operation to more directly influence the recognition process. It has the effect of moving the line dividing perception and cognition toward perception. The perception module is now required to recognise not the entire intention but temporally, spatially, and conceptually smaller elements from which the intention recognition process might be

[58]The interactive simulation community deals with systems that include some human element. Typically these systems are designed specifically for training, rehearsal, or planning and include the human as an active participant in the system. Constructive simulation deals with systems that have no human component. Often these simulations are used for operational analysis and explore large parameterised spaces through monte-carlo or other operations research simulation techniques.

developed. This example illustrates the application of the Sense and Infer pattern from Section 6.2.2.

7.1.5.4 Version Four: Assisted Sense and Infer

The fourth system labels the 'actions' of the pilot to simplify the inferential reasoning required in the previous example. This example illustrates the application of the Assisted Sense and Infer pattern from Section 6.2.3.

7.1.5.5 Version Five: Hybrid

Version five presents the Hybrid pattern from Section 6.2.1. As the name implies this is a pattern that results from a combination of the Ecological Recogniser and the Sense and Infer patterns.

7.1.5.6 Version Six: Clairvoyant

Version Six of the VAS describes the system design that results from the application of the Ecological Recogniser pattern from Section 6.2.6. Because of the wide variety of implementation possibilities with this pattern four different variants of this pattern are described.

7.2 Six Architectural Variants

This section provides an architectural design description for each of the six variants of the VAS. Because the architectural patterns are fundamentally suited to different types of systems the variants are chosen to best illustrate the benefits offered by the patterns. This introduces an artificiality: the systems were chosen to present the patterns rather than the patterns being chosen to best suit the systems. It would have been possible to invent a set of requirements to justify pattern selection but that is potentially misleading and a more informative approach is to simply present the system and then describe the strengths and weaknesses of the architecture.

These architectural descriptions compose three sections:

1. an overview of the particular system;
2. A description of the major architectural components with an accompanying diagram and an indication of the interactions between the components
3. A discussion of the implications of the architectural pattern for the system.

System designs appearing in this part of the thesis are described both in natural language and with the UML. This demonstrates the utility of the UML in assisting in the design of agent systems. The UML is a modelling language and does not presuppose

or specifically support a methodology (despite its close links with OO and the Rational Unified Process) but the UML is rapidly becoming a software engineering design standard for object oriented development [155] and most practicing software engineers and virtually all new graduates will have at least some familiarity with the notation. On occasion the UML notation is adapted from standard use to meet the specific requirements of agent systems development. Efforts have been made to maintain consistency with emerging practices for documenting agent systems [132, 136, 73].

7.2.1 Hybrid

This variant of the Virtual Air Show architecture is an application of the Hybrid Recogniser pattern (See Section 6.2.1 Figure 7.6).

7.2.1.1 System Description

The *Hybrid* variant of the Virtual Air Show is a real-time, human in the loop, flight simulator. The system provides the human user with an out-of-the-cockpit view from a light aircraft and an instrument panel display that allows flight of the aircraft around a simple three dimensional world (See Figure 7.21 for an actual screen shot of the system as it was finally implemented). The user can taxi, take-off, fly around, and land the aircraft in a simple three dimensional world. When taking-off, landing, or flying in the circuit, an agent operating as a *flying instructor* provides the pilot with appropriate advice about actions to take. By way of a short-hand, and to distinguish it from other agents in later variants of the VAS the agent will be referred to as the *Instructor Agent*. As part of its behavioural repertoire the Instructor Agent is required to recognise the intention of the human. The virtual air show consists logically of three components: the flight simulation engine (including the aircraft and environment models); the graphical user interface; and the instructor-agent. Beneath this conceptual description is a software architecture that makes use of pre-existing legacy code for the flight simulation components and integrates two very different technologies in the construction of the Instructor Agent. The finished system will make use of speech synthesis software to allow the Instructor Agent to communicate with the human pilot. The system architecture is shown in the UML diagram of Figure 6.2.

7.2.1.2 Discussion

Flight simulators will nearly always provide some set of physical state data to systems hat must connect to it. Indeed the emergence of the Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) protocols is a very good example of this. Modules are likely to be easier to integrate with flight simulators if they make use of the data defined by these protocols. Unfortunately these protocols do not allow for agent-specific data that might be required to implement either of the *Assisted* patterns or the *Clairvoyant* pattern. This means that, at least for a simulation domain, there is some advantage in opting for the Hybrid, the Ecological Recogniser, or the Sense and Infer pattern.

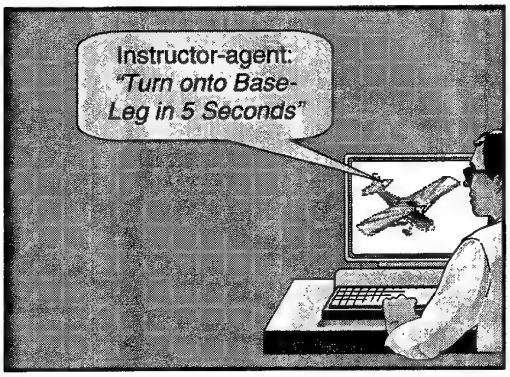

Figure 7.5: The Ecological Recogniser variant is a flight simulator piloted by a human. The Instructor Agent *observes the flight, and by recognising the intention of the human pilot, offers appropriate, timely advice to the user via a speech synthesis utility.*

The adoption of the Hybrid pattern allows for the use of hybrid technologies to implement the detail of the intelligent agent. This has the advantage of allowing the choice of technologies that are well suited to the particular task. In the case of the Instructor Agent the task of recognising *actions* in the streams of aircraft state data is well suited to a pattern matching algorithm whereas the task of reasoning about the meaning of those actions in the context of recognition is well suited to high-level symbolic agent languages.

A serious disadvantage of the Hybrid pattern is the dispersal of the intention recognition functionality into the perception and the reasoning module. Increased design complexity results from the lack of cohesion associated with spreading the functionality. Although a solution like the Ecological Recogniser Pattern of Section 7.2.4 might also make use of hybrid technologies it modularises those technologies and confines them to specific modules of the agent. For example, the *Ecological Recogniser* might well make use of a high level, symbolic agent programming language combined with a network based module for perception, but the intention recognition functionality is confined to the latter module and so there is higher cohesion. The *hybrid* pattern makes use of an architecture that will possibly result in implementations that see two modules, dissimilar in many respects, cooperating to provide the intention recognition functionality. This has benefits if the two processes (perception and reasoning) are different enough to warrant that split but in the context of this thesis the phrase "different enough" refers to the software engineering requirements and not to models of psychology or human characteristics.

7.2.2 Sense and Infer

This variant of the Virtual Air Show architecture is an application of the Sense and Infer pattern (see Section 6.2.2).

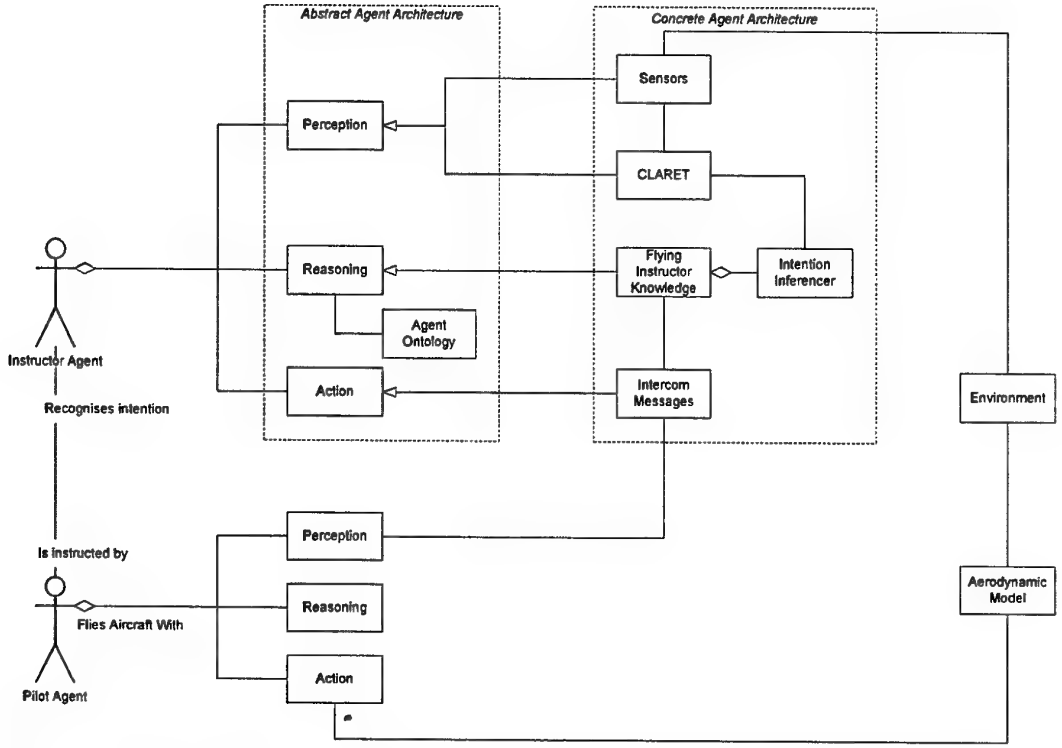

Figure 7.6: The aircraft state *is produced by the* aerodynamic model *and provided to the agent through the simulation* environment. *This aircraft state takes the form of data structures that contain information about the aircraft's geometric position (including its position relative to environmental features) its speed and acceleration and other aspects of its state (is the landing gear up or down, what angle are the flaps set at). These data structures are updated as the aircraft moves.*
The aircraft state is taken by the sensors *and passed to a pattern matcher (indicated here by the CLARET module). The sensors are rudimentary models of the aircraft instruments and the pilots field of view. Together the sensors and the pattern matcher constitute the* perception *module. The pattern matcher processes the state data checking against trained examples searching for patterns in the input data that it can bind to previously trained descriptions of pilot action. If the pattern matcher recognises an action it outputs it to the* reasoning *module of the agent. The reasoning module uses the information about the action as the basis for its reasoning about the intention of the Pilot. The recognised intentions are used as the basis for determining the appropriate advice to offer the pilot.*

7.2.2.1 System Description

This variant of the VAS is functionally similar to the Hybrid variant but employs the Sense and Infer pattern which results in architectural differences. The resulting system will have some of the properties of the system described in Section 2.3. The system architecture is shown in the UML diagram of Figure 7.7. A feature of the Sense and Infer pattern (and the Ecological Recogniser and the Hybrid) is that they can function equally well if the pilot is a human in the loop or an agent. Figure 7.7 shows the system with an intelligent agent as the pilot but there is no reason why the Instructor Agent design would change if the Pilot Agent were replaced with a human pilot. This idea has been explored in military simulations [78].

The purpose of a system with an agent for Instructor and Pilot (outside of demonstrating the themes of this thesis) is unclear though it is possible to conceive of a pilot-agent filling a role testing the previous the human-based system over a wide range of scenarios.

The resulting is a system in which the instructor-agent will guide the 'pilot-agent' around the circuit. The similarities between the two systems are obvious, and a possible solution to developing the third system would be to reuse the first in its entirety but construct a pilot-agent that mimics the behaviour of the human. Research has been published that details ways that this might be implemented with inductive learning [157] and using a BDI agent as was used for the instructor agent.

7.2.2.2 Discussion

The Sense and Infer pattern encapsulates the intention recognition functionality inside the Instructor Agent. This uncouples the Instructor Agent from the Pilot (as is the case with the Assisted patterns or the Clairvoyant). The inference process that results in recognition occurs completely inside the agent's reasoning subsystem. Any intermediate results are available to influence other aspects of the Instructor's behaviour. This allows the agent to reason in more sophisticated ways about the nature of the pilot's intentions making it is simpler to provide for interactions between intention recognition and other agent functionality.

The architecture is relatively simple but because all of the responsibility for intention recognition now lies within the agent's reasoning module the Instructor Agent reasoning becomes more difficult to design. This has the follow-on effect of compromising the agent's *knowledge-level*. Recalling Chapter 1 it was stated that an agent should, desirably, make use of *knowledge-level* concepts. This pattern requires the agent to reason with lower level aircraft state data.

Performance issues are possibly problematic for this architectural design. Even though the set of possible intentions to be recognised is small there is a significant processing overhead in deducing the actions that have been undertaken by the pilot from the aircraft state data. There is enough variability in many of the seemingly simple manoeuvres to create significant challenges in codifying the recognition as a set of rules.

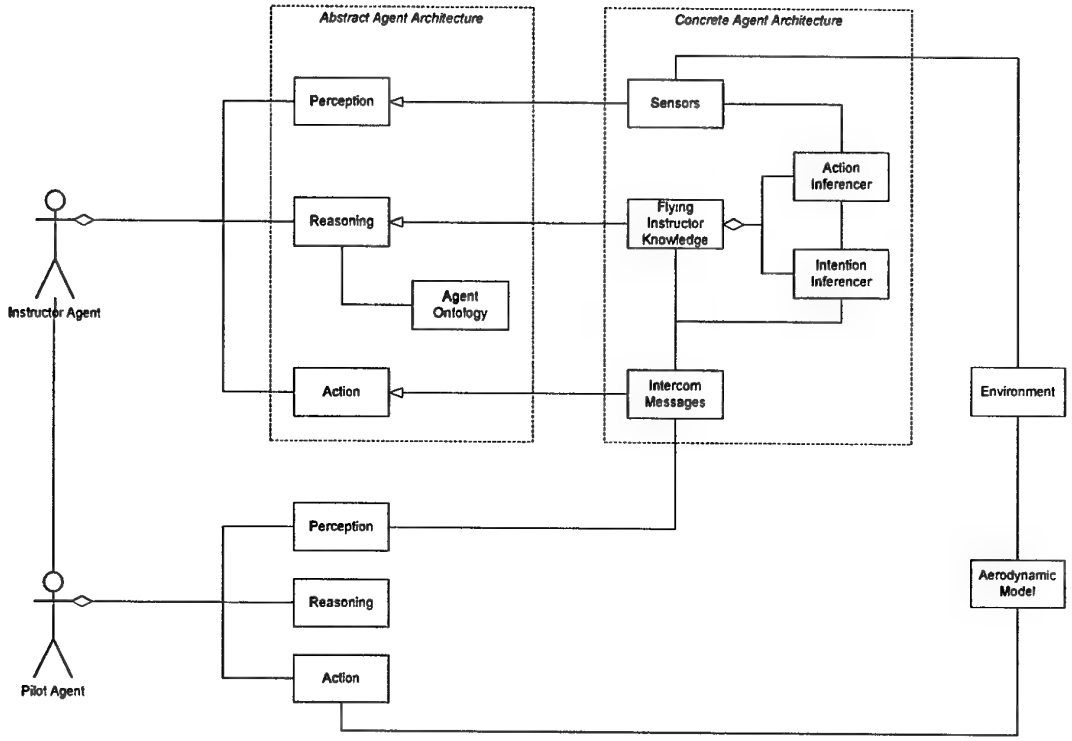

Figure 7.7: The aircraft state *is produced by the* aerodynamic model *and provided to the agent through the simulation* environment. *This aircraft state takes the form of data structures that contain information about the aircraft's geometric position (including its position relative to environmental features) its speed and acceleration and other aspects of its state (is the landing gear up or down, what angle are the flaps set at). These data structures are updated as the aircraft moves. The* update rate *is a function of the particular aerodynamic model and for most flight simulators is between 10 and 30 Hz.*
The aircraft state is taken by the sensors *and passed to an* action inferencer. *The action inferencer and the subsequent* intention inferencer *are both modules of agent reasoning. The reasoning module uses the information about the intention to pilot to provide the necessary advice.*

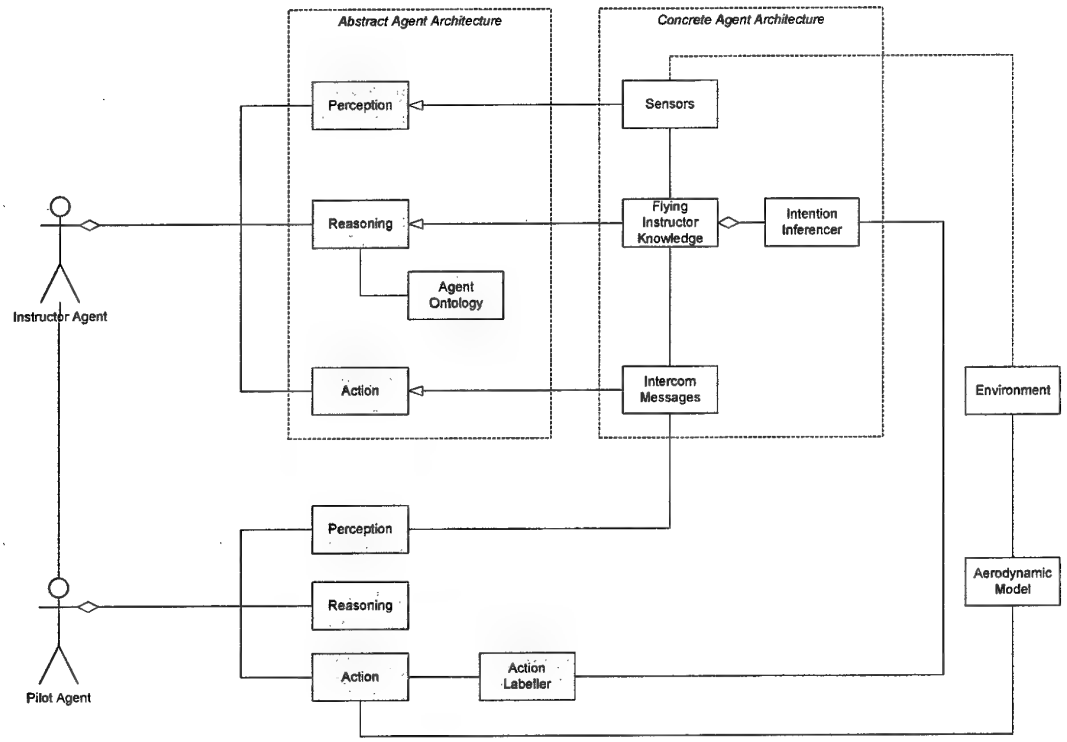

Figure 7.8: The action state *is produced by the* action labeller *and provided directly to the agent. The action state is taken by the Agents* reasoning *module—in this case the* intention inferencer *where it is used as the basis of reasoning about the intention of the Pilot Agent. If the pattern matcher recognises an intention it outputs it to the* reasoning *module of the agent. The reasoning module uses the information about the intention to pilot to provide the necessary advice.*

7.2.3 Assisted Sense and Infer

This variant of the Virtual Air Show architecture is an application of the Assisted Sense and Infer pattern (See Section 6.2.3 and Figure 6.4).

7.2.3.1 System Description

This variant of the system is identical to that described in Section 7.2.2 with the exception that it will make use of an addition to the Pilot Agent that will label the agent's *actions* so that these might be used by the Instructor Agent. The system architecture is shown in the UML diagram of Figure 7.8.

7.2.3.2 Discussion

This version of the VAS represents a half-way point between the extremes of the Clairvoyant and the Sense and Infer pattern. As a VAS architecture it tends to acquire the advantages of both at the same time avoiding the problems of either. The attaching of

labels to actions seems intuitively less subjective. Though the reason for an action might be subjective the action itself is likely to be more objective. Placing labels on things that are not likely to be interpreted differently by an agent under different internal states removes many of the potential modelling issues. In this case the actions that were labelled are obtained from the internal processing of the pilot agent.

Using inference in the reasoning model to provide the recognition of intention data based on a knowledge of the actions is a more appropriate design. The Assisted Sense and Infer benefits more from the *assistance* than the Assisted Ecological Recogniser does.

The task of inferring intention based on a knowledge of the actions is sensible for a Pilot Agent where a more abstract representation of action is available but for a human agent it is less practical. One possible variation of this system is to replace the Pilot Agent with a human pilot and to use the mouse and keyboard as the representation of the action states. This variant is explored further in Section 7.2.5.

The architecture has linked the intention recognition process with the pilot actions. This has coupled the two agents and a modification to any particular actions that might be used by the Pilot Agent require corresponding changes to the Instructor Agent.

7.2.4 Ecological Recogniser

This variant of the Virtual Air Show architecture is an application of the Sense and Infer pattern (see Section 6.2.4). The application of this architectural pattern is explored in more detail in Section 7.3.

7.2.4.1 System Description

This variant of the VAS is the one taken to implementation in Section 7.3, consult that section for more details. This variant of the VAS is similar in functionality to the Hybrid variant. The Hybrid makes use of its perception module to recognise actions which are then processed into intentions by the reasoning module. The Ecological Recogniser takes the jump from state data to intentions in a single step and voids the need for any reasoning about intention. The architecture shown in Figure 7.9 shows the major components and their interactions. Understanding the design of intention recognition is explained by tracing the data flow through the system whilst focussing on the central role that *perception* plays in this design.

7.2.4.2 Discussion

A significant software engineering advantage lies in removing the need for an agent to be explicitly provided with a capability for intention recognition inside its cognitive architecture. Instead a sophisticated pattern matching algorithm allows the intention-instances to be demonstrated rather than explicitly coded. Thus the task of specifying the rules that govern intention recognition becomes a demonstration task and not a specification task.

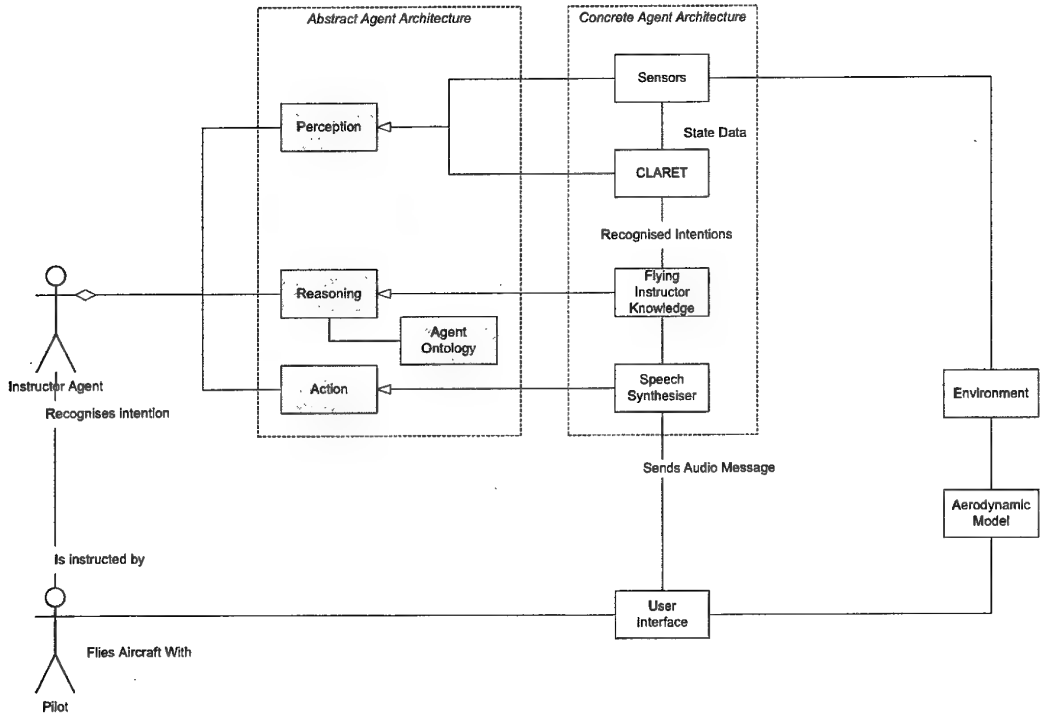

Figure 7.9: The aircraft state *is produced by the* aerodynamic model *and provided to the agent through the simulation* environment. *This aircraft state takes the form of data structures that contain information about the aircraft's geometric position (including its position relative to environmental features) its speed and acceleration and other aspects of its state (is the landing gear up or down, what angle are the flaps set at). These data structures are updated as the aircraft moves. The* update rate *is a function of the particular aerodynamic model and for most flight simulators is between 10 and 30 Hz.*
The aircraft state is taken by the sensors *and passed to a pattern matcher. The sensors are rudimentary models of the aircraft instruments and the pilots field of view. Together the sensors and the pattern matcher constitute the* perception *module. The pattern matcher processes the state data checking against trained examples searching for patterns in the input data that it can bind to previously trained descriptions of pilot intention. If the pattern matcher recognises an intention it outputs it to the* reasoning *module of the agent. The reasoning module uses the information about the intention to pilot to provide the necessary advice.*

A modelling advantage lies in the combination of a high level symbolic reasoner to manage the cognitive aspects of the agent and a sub-symbolic machine learning algorithm to manage the sub-cognitive aspects associated with perception and recognition. By applying appropriate technologies to these two distinct parts of modelling human behaviour a more realistic total model results [120].

A sophisticated model of perception that maps the data in the environment into that required by the reasoning module of the agent removes the need to modify either the environment or to add an inferencing capability to the agent. Thus although this variant has a highly complex perception subsystem the environment could be left unmodified and the agent design could be implemented without needing to consider the intention recognition problem. The system is independent of the embodiment of the pilot–human or agent, it makes no difference. This feature is shared in common with the Sense and Infer pattern and the Hybrid. The two *Assisted* patterns can potentially operate with human pilots but the design is both conceptually and practically more difficult to conceptualise.

A perception module that must pattern match against the incoming data streams will not be a simple system. It is computationally feasible in a single agent system such as the one demonstrated here (see the implemented example in Section 7.3) but in more complex multi-agent systems (many of which employ dozens of agents) the overhead associated with the greater number of agents will be problematic. When a machine learning system must be trained with examples, as is the case with CLARET (again refer to the implementation), there is a resulting validation challenge that might not be easy to solve. For non-critical systems this might be acceptable but for systems that require a high degree of robustness this might be true.

Integrating the pattern matching software with reasoning module will present integration issues. The likely mix of technologies, the different data abstractions and the control of processing will combine to create a complex system integration.

By removing the intention recognition functioning from the agent cognitive reasoning and placing it into the perception module there is little opportunity for the agents reasoning to influence the recognition process. Solutions for this specific architecture have been proposed elsewhere and this is a problem that has been encountered by many other AI systems.

7.2.5 Assisted Ecological Recogniser

This variant of the Virtual Air Show, is an application of the Assisted Ecological pattern (See Section 6.2.5 and Figure 6.6).

7.2.5.1 System Description

This variant of the VAS is similar, as the name suggests, to the Ecological Recogniser (Section 6.2.4). The Assisted Ecological pattern makes use of the *action state* of the *intending agent* rather than the environment state. In the context of the VAS this implies that the pilot's actions are the source of the perception and not the physical state of the aircraft. The architectural design reflects this change by providing the perception module

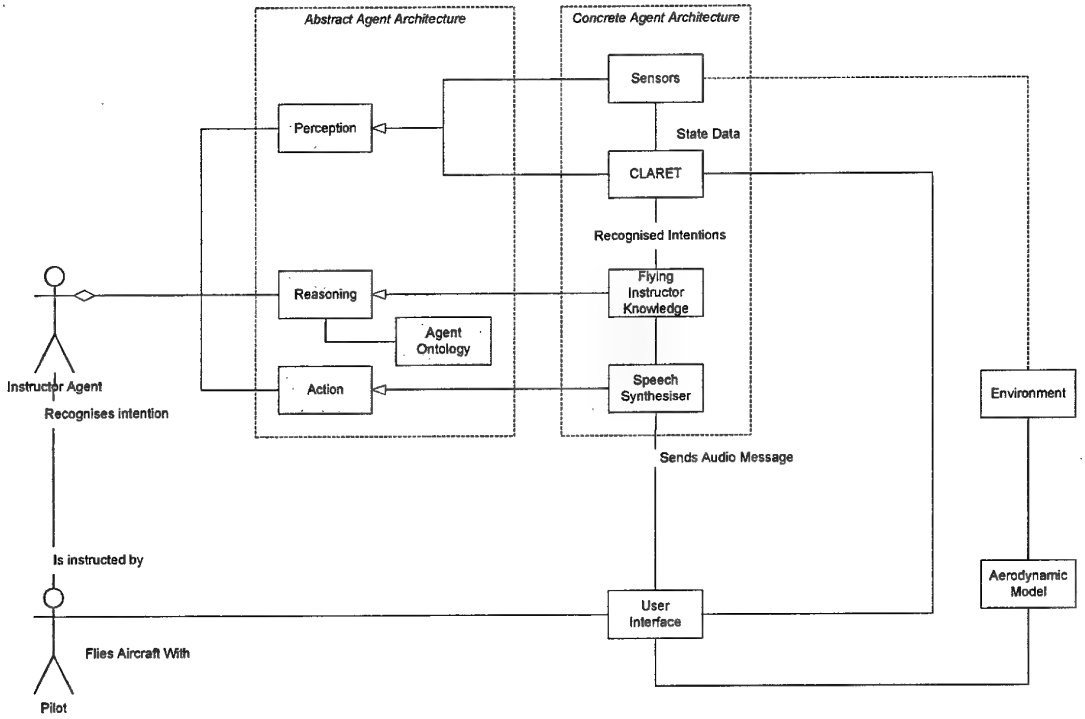

Figure 7.10: This variant of the VAS allows the Instructor Agent direct access to the actions of the pilot. These actions are the throttle and stick movements in the case of a real pilot or to a more suitable abstract representation in the case of an intelligent agent pilot. of the pilot. Inaccessible in the real world, the time histories of these data may also be indicators of the pilot intent. Preliminary experiments indicate that, in fact, CLARET performs poorly with these data (see Section 7.3). The state of the controls, the mouse and keyboard, inputs are sensed and passed to a the pattern matcher. The state data are processed for the patterns of inputs that can be bound to recognition *of intention. If an intention is recognised it is announced to the Reasoning module. The reasoning module uses the information about the intention to pilot to provide the necessary advice.*

with a data stream from the mouse and keyboard that are used to fly the aircraft. In every other respect the architecture is the same. Quite simply then this system can be considered as a flight simulator where an agent uses a perception module to recognise the pilots intention by observing the use of the controls. The system architecture is shown in the UML diagram of Figure 7.10.

7.2.5.2 Discussion

Knowledge of the agent's actions (in this case the stick and throttle positions and any mouse and keyboard commands) are easily accessible within the existing simulation with very little modification. This combined with the reduced data set provided to the pattern matcher (eight variables are required to characterise the control inputs compared with as many as twenty to characterise the aircraft state) promises to simplify the processing of the pattern matcher and improve performance.

Preliminary experiments indicate that the recognition process performs no better with time histories of the control inputs than it does with the aircraft state. In several instances it significantly under-performed. Investigating this is beyond the scope of this thesis but it almost certainly a particular property of the flight simulation domain. Intuitively the position of the aircraft would seem better indicators of the intention of the pilot than the control movements that led to the manoeuvring. There may be many instances where the feeding the action data into a pattern matcher may result in substantial benefits[59].

There is no credible argument from cognitive modelling for adopting this architecture. Except for the implausible case of a flying instructor actually watching a students throttle and stick positions rather than the instruments and the environment. Unlike the Ecological Recogniser pattern which can be justified by reference to ecological psychology or naturalistic decision making, the Assisted Ecological Recogniser pattern is difficult to justify from a standpoint other than software engineering pragmatism.

Consider a variant of this version with an intelligent agent replacing the human pilot. If the pilot agent had an architecture like that shown in Figure 7.11 then the possibility exists for a more abstract set of actions to be sent to the pattern matcher.

7.2.6 Clairvoyant

This variant of the Virtual Air Show architecture is an application of the Clairvoyant pattern (see Section 6.2.6 and Figure 6.7).

7.2.6.1 System Description

This variant of the VAS is another example of the Pilot Agent based system. This example of the VAS requires the pilot to allow direct access to a representation of its intentions. This direct access effective short-circuits the intention recognition process. This is a strong example of the simplicity that can be achieved through a reconsideration of the engineering of agent systems.

This variant, the simplest of all, reduces intention recognition to simple communication. In practice this might be implemented in a number of different ways but fundamentally there is no need for the instructor agent to perform any reasoning about intention at all. The pilot simply informs the instructor of his/her current intention.

The system architecture is shown in the UML diagram of Figure 7.12. The architecture is so simple that it requires almost no explanation. In the detail of the implementation there is scope for interesting variations on the architectural theme.

[59]Experiments, outside the scope of this thesis, conducted with the implemented system described in Section 7.3 suggest that if the pilot flying the flight simulator is not skilled then intention recognition based on the control inputs offers better performance, but if the pilot is skilled, the aircraft trajectory is preferred. Indications are that a carefully selected combination of the two data sources results in better performance by the pattern matcher. Preliminary results have been published elsewhere [72].

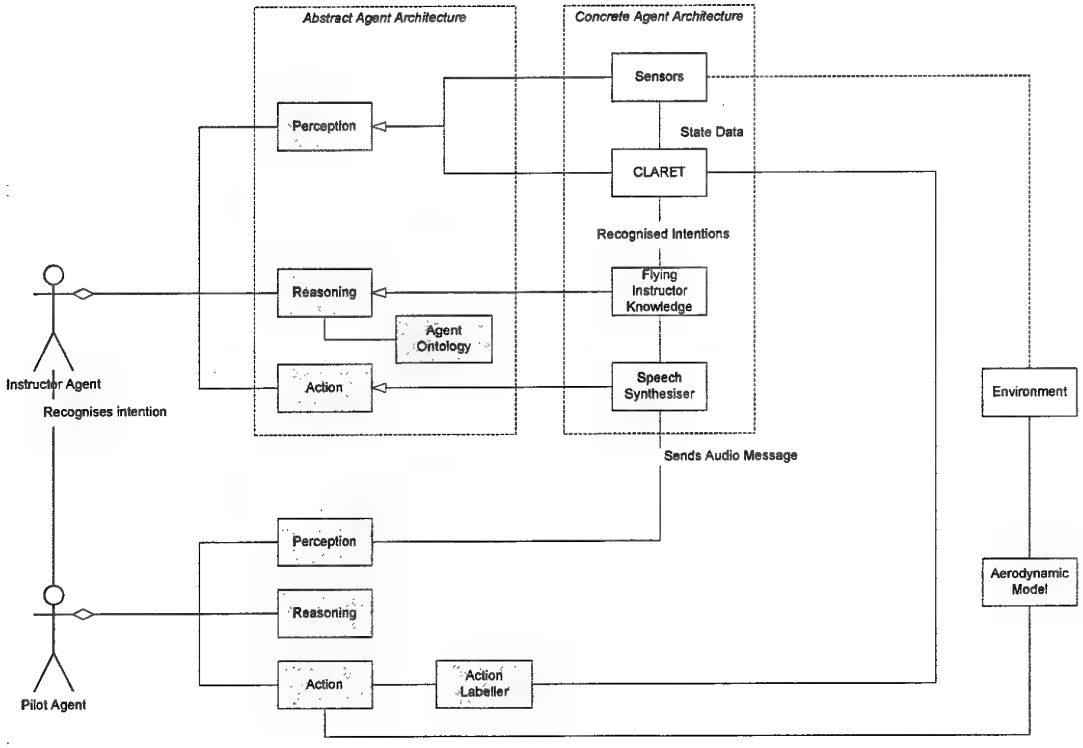

Figure 7.11: A variant of the Assisted Ecological pattern shown in Figure 7.10 replaces the human pilot with an agent. If the Pilot Agent *implements an architecture like that shown then a more abstract representation of action than is available. This supports a more appropriate use of this pattern.*

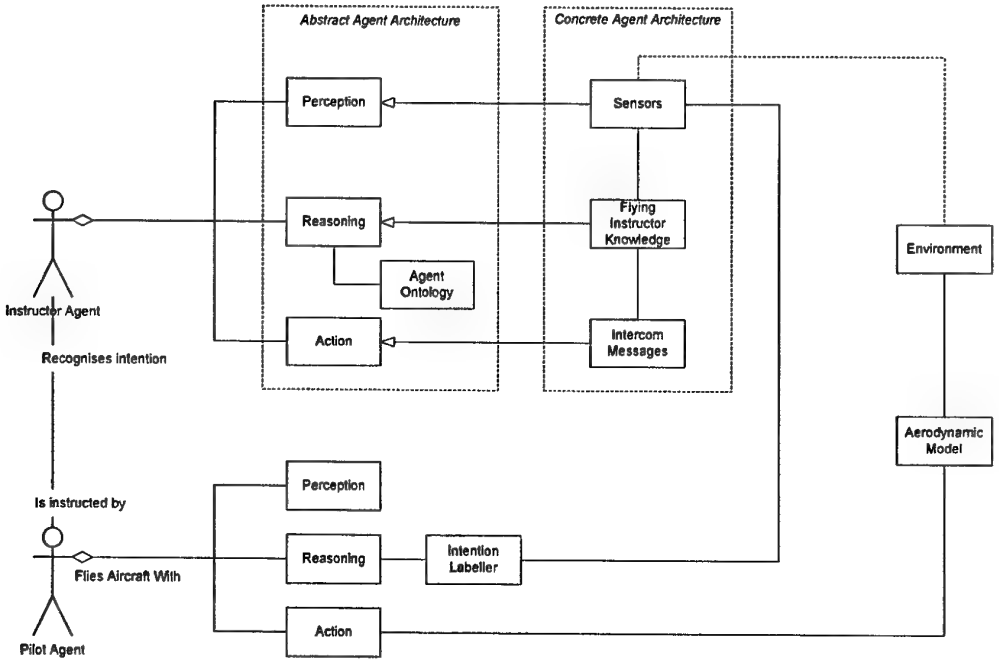

Figure 7.12: The Clairvoyant architecture is almost trivially simple. The Pilot Agent is provided with an intention labeller. *The Instructor Agent is provided directly with an indication of the intention of the Pilot Agent.*

7.2.6.2 Discussion

This is the simplest implementations of intention recognition. One agent can sense the intention of the other directly. Conceptually this is by reading a label that the agent attaches to itself, in practice it might be achieved by directly accessing the internals of the agent, receiving a broadcast transmission, or many other options. The Clairvoyant pattern will clearly provide the least computational effort. The solution may not be *intuitive* from a cognitive science perceptive but it is an obvious solution if simplicity is required. The solution is compatible with communication. An extension of the VAS that required a dialog between the pilot and the instructor about the nature of the pilot's intention would be compatible with this pattern. Many variants of the basic pattern are possible providing various possibilities with different properties. The system can scale almost indefinitely. The computational requirement is so low as to render any overhead inconsequential. The agents are very strongly linked. It is difficult to model errors in recognising intention because the existence of intention originates with the pilot. It is similarly difficult to model delays in recognising intention and other human performance aspects. Intention recognition is inherently subjective. When an agent displays a single indication of its intention all subjectivity, chance of error, and the ability to model the effect of the internal state of the recognising agent on the recognition process is lost. If the Instructor requires detail about the intention of the pilot it must either be provided (violating the basic premise of the pattern) or it must be inferred compromising many of the stated benefits of this pattern.

7.2.7 Architecture Summary

This section presented a set of six architectural design for the Virtual Air Show. Figure 7.13 summarises the architectures that result from the application of the patterns. The variation in the attributes of these architectures is large. Together they span a large design space that provides the designer with a variety of options.

7.3 Description of an Implemented System

This section discusses pertinent details of the complete implementation of the Ecological Recogniser variant of the VAS. The development of this system, although it was a technology demonstrator and likely to have no application outside of this research, was undertaken with a strong focus on software engineering. Every development stage was comprehensively documented. Only a small subset of the software documentation is presented here and the following sections provide a glimpse into the analysis and design process with a bias toward those aspects of the system that are concerned with intention recognition.

This entire section is not central to the core of the thesis and many of the details have been published elsewhere [72, 71]. The details are presented here because they elaborate some of the claims on the periphery of this research and because the existence of an implemented version of this system is necessary to establish the technical feasibility of the design patterns.

Section 7.3.1 presents a very brief account of the manner in which the UML might be used to document the intentions of intelligent agents in a system.

The core of the flight simulation system is fairly routine software running an aerodynamic model sourced from a military flight simulator and an Open-GL visual display system that provides a three-d out-of-the-cockpit view of the environment and a representation of a generic cockpit with the necessary instruments and controls (See Fig. 7.21).

7.3.1 Specifying the Intentions to be Recognised

Specifying the requirements that are to be recognised is an important part of modelling intention recognition for intelligent agent systems. A possible approach to analysing and the requirements to be specified was canvassed briefly in Section 5.2.4 and an example of the use of the UML as an aid to documenting intentional agent behaviour is included here for completeness. This specification was undertaken as part of the software engineering of the Ecological Recogniser variant of the VAS and is included here as an aside to the central core of the thesis.

The analysis phase of software development is concerned with advancing the understanding of the requirements that specify the system to be built toward an architectural and then a subsystem design. A detailed look at the analysis phase of the development of the VAS is outside the scope of this thesis but it is important to demonstrate that the types of software processes necessary for selecting a suitable design pattern are available.

	Input to perception module	Function of perception module	Output of perception module/input to reasoning module	Function of reasoning module	Output of reasoning module
Hybrid	Aircraft state	Process into pilot action	Pilot action	Process into pilot intention and advise pilot	Advice to pilot
Ecological	Aircraft state	Process into pilot intention	Pilot intention	Advise pilot	Advice to pilot
Sense and Infer	Aircraft state	Transmit unchanged	Aircraft state	Process into pilot action and intention and advise pilot	Advice to pilot
Assisted Ecological	Pilot action	Process into pilot intention	Pilot intention	Process into pilot intention and advise pilot	Advice to pilot
Assisted Sense and Infer	Pilot action	Transmit unchanged	Pilot action	Process into pilot intention and advise pilot	Advice to pilot
Clairvoyant	Pilot intention	Transmit unchanged	Pilot intention	Advise pilot	Advice to pilot

Figure 7.13: A comparison of the architectures that result from the application of the 6 patterns to the Virtual Air Show

Analysing the requirements of an agent system can be undertaken in many ways and there are several well reasoned approaches to Agent Oriented Analysis that are emerging [185, 182]. The approach presented here is to adopt the basic notations of the UML and to apply the standard analysis techniques that are associated with the application of the UML to the agent components. This allows a fundamentally similar triplet of tools, techniques, and methodologies to manage both the agent and non-agent aspects of the software. Many developers of agent oriented methodologies appear to underestimate the interplay between agent and non-agent components of their systems and consequently fail to provide the design time support for modelling agents and environments in an integrated manner. Approaches like Prometheus [182] and Gaia [185] may be appropriate for agent exclusive systems but there is little support given for developing systems were the design must trade-off functionality between agent and non-agent components. This problem is being addresses in a bottom-up manner with those looking to integrate agent and non-agent software development at the tool and language level (the AUML being an excellent example) and those looking at integration through methodology engineering (an example is the work of Juan et. al. [89]).

Three types of analysis were performed: a standard use case analysis was used to help specify the requirements and was analysed to develop the system design; an intentional analysis (in practice a modification of the agent use case analysis suggested by Heinze, Papasimeon and Goss [73, 59]) was used to document the intentions of the actors; and finally an intention recognition analysis documented the intentions that were to be recognised. As suggested in Chapter 5 the second of these is useful even if intention recognition is not required because intentional descriptions of agent behaviour are an excellent way of documenting the required behaviour.

The complete software documentation is both voluminous and unnecessary for illustrative purposes so only a small indicative subset of the analysis undertaken is presented here. This section documents the high-level requirements of the system by first elaborating a primary scenario that is decomposed and described in terms of the intentions present in the system. These intentions are documented both with high level descriptions, decomposition where appropriate and finally by describing the characteristics that can be perceived by an agent. These description can, like the intentions themselves be decomposed into increasing detail as appropriate.

This distinguishes this type of analysis, from the traditional use case analysis where a number of scenarios are analysed and the resulting use cases indicate some grouping that characterises the functionality that system provides to the user(See Fig. 7.14).

This type of analysis concentrates on providing descriptions of the intentions of the agents and the actors in the system. Though this is used here for documenting the intentions that will be subject to a recognition process there is broader application for this type of analysis in the construction of agent systems.

As discussed in Section 5.2.4 there are methods available for gaining insights into the intentions of participants in a scenario. A detailed discussion of these is not relevant here and it will be assumed that some appropriate technique can be employed.

The UML is commonly used to document the use cases and it has many of the features required to document intentions at this level. Details for each of the use cases can now

be added. This might be in plain English as below or any of the UML diagramming techniques deemed suitable can be employed.

7.3.1.1 Example Scenario

In order to illustrate, capture and elaborate the requirements and the intentions in the system a number of scenarios will be presented, described, analysed and documented. The purpose of this analysis is to allow the determination of exactly what functionality is requires and what intentions are present and by what features they might be recognised.

Consider a scenario where the pilot takes off and flies a circuit. The sequence of actions that is occurs during a takeoff, a circuit, and a landing is as follows:

1. Pilot - Take-off and Climb: The aircraft is powered up and flown into the wind along the runway. At an appropriate speed the pilot pulls back on the stick to raise the nose of the aircraft and initiate the lift-off. Following this is the initial climb to a safe manoeuvring height. During this climb the landing gear will be retracted and the flaps may be returned to their undeployed setting.

2. Instructor Agent - Observes pilot.

3. Pilot - Turn to Crosswind Leg: At a predetermined height, generally 500 feet above ground level (AGL) the aircraft is turned through ninety degrees to the left[60]. This turn is a gentle turn, typically twenty degrees angle of bank. After this turn is complete the fuel pump, trim and flaps are checked and the climb is continued until circuit height is reached, usually 1000 feet AGL.

4. Pilot - Crosswind Leg: The crosswind leg requires the pilot to fly at constant altitude at right angles to the runway. Because the wind is blowing from the right hand side of the aircraft it may be necessary to slightly angle the aircraft into the wind to remain on course.

5. Pilot - Turn to Downwind Leg: The turn to downwind leg is a medium turn at constant circuit height (1000 ft AGL).

6. Pilot - Downwind Leg: The downwind leg is flown at constant altitude and speed. During the downwind leg the pre-landing checks are commenced and a radio call is made.

7. Instructor Agent - Advice about turning onto Base Leg: The instructor recognises that the pilot is intending to turn onto the base leg. The instructor agent provides advice about when to turn. "Commence the turn onto base-leg in about 5 seconds".

8. The pilot turns onto Base Leg.

9. the Pilot turns onto the final Leg and lands.

[60]Left hand circuits are most commonly used at airports. Occasionally, when geography, traffic, weather or other conditions apply circuits may be flown to the right.

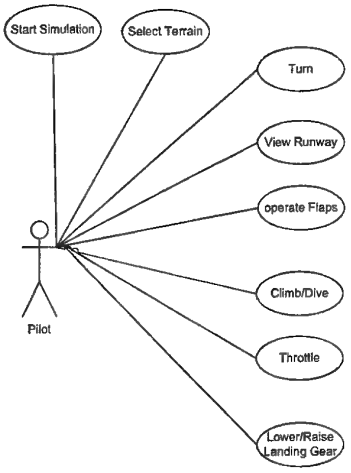

Figure 7.14: Part of the use case model for the Virtual Air Show. The traditional use case model describes groups of functionality that the system provides to the use. During a use case analysis the focus is on the functionality provided to the user. During the intentional analysis shown in Fig. 7.15 the emphasis is on the intentions of the agents and actors involved in the scenario.

7.3.1.2 Use Case Analysis

A standard use case analysis documents the functionality that is afforded the user of the system [155]. This use case diagram groups the functionality into the use cases shown and allows them to be documented using the diagramming techniques of the UML. Any further detail of the use case analysis is beyond the scope of this thesis but an example use case diagram is presented here for comparison with the intentional analyses that follow (see Figure 7.14).

7.3.1.3 Intentional Analysis

A standard use case analysis documents the functionality that the system must provide (Fig. 7.14). An intentional analysis documents the intentions of the actors and agents (See Fig. 7.14). The UML is an appropriate language for documenting agent intentions though in practice the developer can select any appropriate technique or language.

For each intention there might be possible variations and selected courses of actions based upon the particular circumstances. Conceptually this corresponds to a *possible worlds* view of intention as described by Rao [149] but is also consistent with standard approaches for documenting use cases when analysing scenarios [155].

The analyst considers the *intentions* of each of the actors and the documents them in the style of use cases. As shown in Figure 7.15 these intentions can be structured into sub-intentions. In this case the primary intention is to 'fly a circuit' with sub-intentions to 'climb', 'fly the final leg' etc. This simple documentation provides the first specification at the intentions that will be observed in the finished system. It might appear similar to

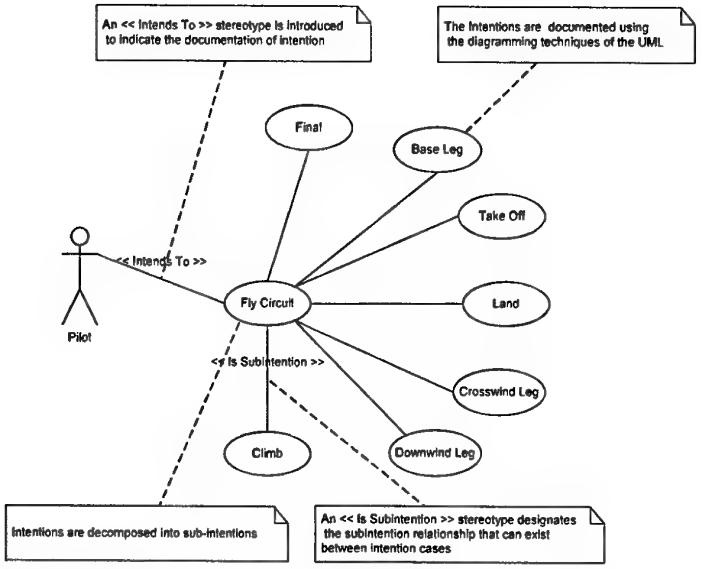

Figure 7.15: Part of the model of the intentions of the agents in the VAS. Because the system under construction is a simulation there is a close similarity between this intentional description of the system and a domain model of the reality that is being simulated. The difference between an intentional description of the domain and the model of the executing simulation is related to the simplifying modelling assumptions made in developing the simulation. The intentional analysis of a simulation system can be considered a realisation *of the real world that is simulated. The distinction is an important, if subtle one, and the constructs of the UML are adequate for documenting both.*

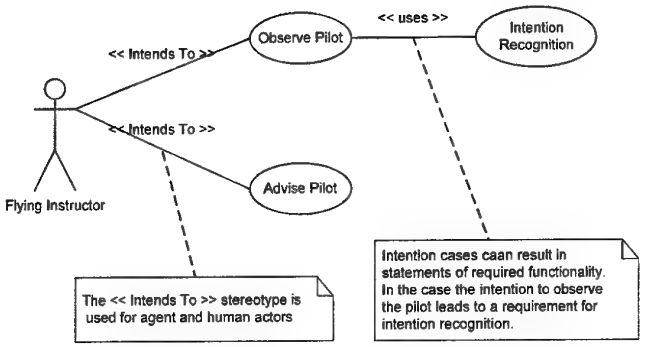

Figure 7.16: Part of the agent intention model for the Virtual Air Show. By treating the agent as an actor that interacts with the system in much the same way as actors traditionally do in a use case analysis it is possible to document and analyse the interactions between the agent and the system.

a use case analysis but the emphasis is on documenting the behaviours observed as the system functions.

A textual description of the intention is added to provide context and detail. Only three are included here:

Downwind Leg The "downwind leg" intention will be manifested by a period of straight and level flight at 1000ft altitude at a certain geometric location with respect to the runway. Primarily it will be parallel to the runway and offset to the left by a distance of a few hundred feet and be flown in the direction of the wind. During the downwind leg the flaps may be set for the descent into the landing and the mixture mill be set to rich. If the course changes toward or away from the runway it is a strong indication that the circuit is being departed.

Turn Base Leg The intention to turn onto the base leg follows from the downwind leg. Once the runway threshold passes abeam of the aircraft the turn will be approaching. There might be a decrease in altitude and the turn will be made when the touchdown point is 30 degrees to the rear of the aircraft. If the wind is strong this turn will be made earlier.

Advice about when to turn onto base leg The instructor agent recognises the pilot's intention to turn onto the base leg and must provide advice about when to commence the turn. Possible confusion exists because the pilot might be intending to depart the circuit by turning early and flying across the top of the runway or extending downwind.

These textual descriptions may be adequate for documenting the early stages of development but a structured decomposition of the intentions is a necessity for making design decisions about intention recognition. Intentions are decomposed into actions. The effects that those actions are expected to have in the world are also documented (See Figure 7.18). This intentional analysis in synonymous with the agent behavioural modelling of Goss et. al. [59].

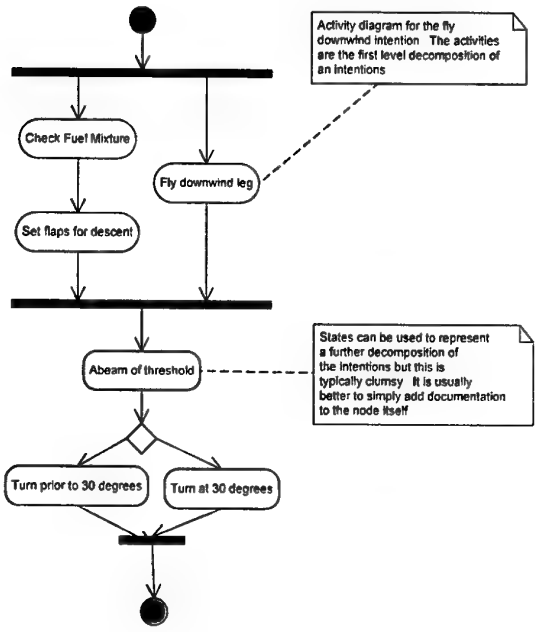

Figure 7.17: Activity Diagram for the Turn to Base *intention.*

7.3.1.4 Intention Recognition Analysis

Each of the intentions that must be recognised is now documented as shown in Figure 7.18. This documentation is based around an elaboration of the features that might be useful in performing intention recognition but should also include any constraints or requirements on the nature of recognition.

7.3.1.5 Ontology

As the concepts of interest are emerge during the intentional analysis it is possible to develop an increasingly detailed view of the ontologies that characterise the system. This is the first estimate of the concepts that the agent must reason about and they arise out of a consideration of the requirements of the agent that must perform the recognition. In a development driven entirely by the requirements of the agent's reasoning these would become the specification that the rest of the system must meet. If there is flexibility in the design process then the specification of this ontology is an iterative process that will arrive at a compromise between the requirements of the agent and the requirements of the rest of the system. Standard shared ontologies is a strong example of the requirement for an agent to reason about a certain set of concepts. Several researchers have proposed extensions to the UML for ontology modelling [36] and so the UML is used for here for documenting the ontology.

The component of the environment ontology relevant to intention recognition is represented with the UML as shown in Fig. 7.20. These specifications of the important concepts in the system represent the form of the knowledge available in the system and the form required by the agent (Secc 7.19).

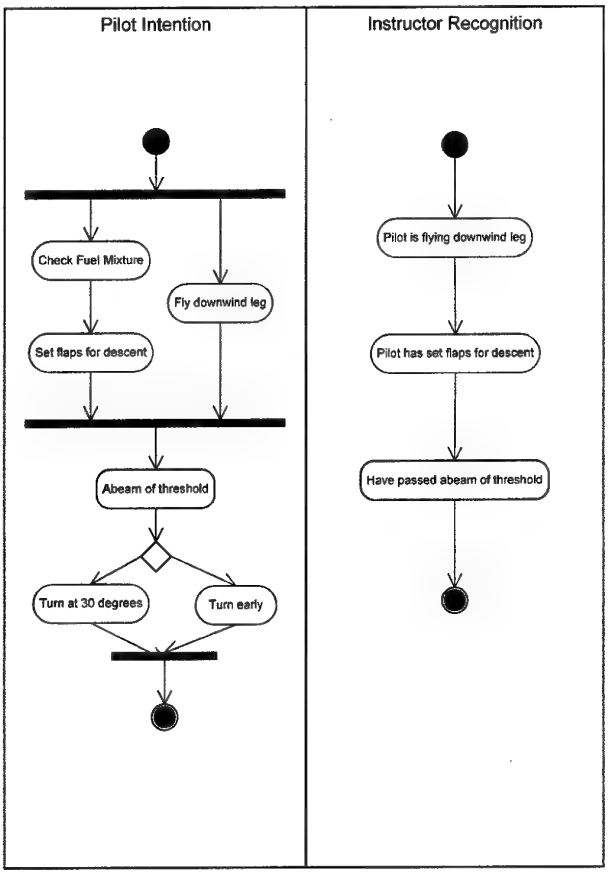

Figure 7.18: Detail of the description of the "Turn to Base Leg" intention. Note that the description includes the features that characterise the intention as viewed by the instructor. Determining just how these features are to be made available to the agent, in what form, and how they are perceived is a key part of the process of engineering intention recognition.

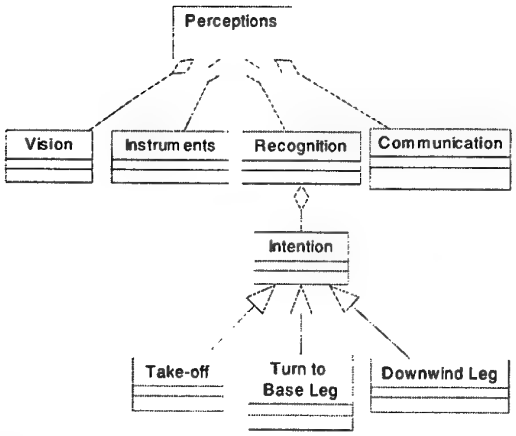

Figure 7.19: The agent ontology for the purposes of design. This captures the concepts that the agent reasons with that are of concern in designing the interface between the agent and the rest of the system. Though used here as a design-time aid in more complex system it might also have run-time utility in providing well designed ontologies that can be switched in and out in response to events.

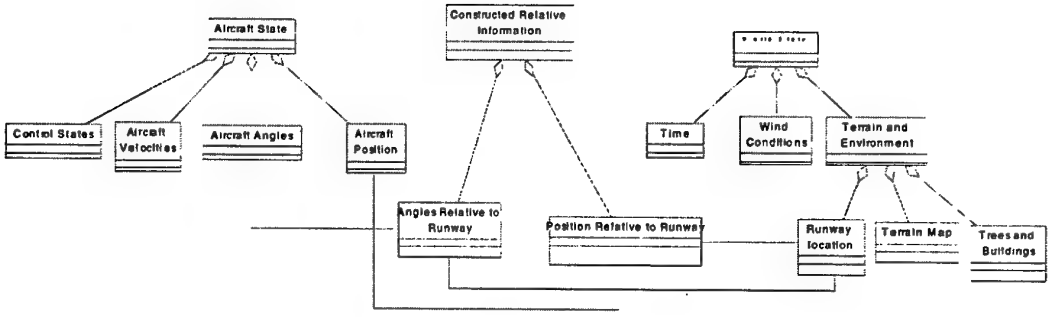

Figure 7.20: The environment ontology is a summary of the relevant data available to the agent. In larger, more complex systems this ontology is an aggregation of the ontologies of the various components that comprise the system. In the case of the virtual air show this is unnecessary.

Perception has the task of mapping from one system ontology to the agent ontology. The agent ontology is retained as a part specification of the agent interface. In more complex systems, where communication and perception results from many different sources these ontologies might be loaded and unloaded at run time to configure the agent for certain activities.

7.3.2 The Implemented System

This section provides relevant details of the implemented system. Further details of this particular implementation have been published elsewhere [72] and explain in more detail than is warranted here the relationship to military simulators. Other variants of the VAS are presented in the following sections without implementation details as they are unnecessary for the presentation of the application of the design patterns. See Figure 7.22 for an embellished architectural diagram.

7.3.2.1 Hardware

The system was implemented on a 4 CPU 200MHz Silicon Graphics server running IRIX6.3 and running a VTX graphics subsystem.

7.3.2.2 Environment, Aerodynamics and GUI

The model of the atmosphere, the physical environment and the associated graphical representations were reused from an existing flight simulator—FSIM. Detailed descriptions of the implementation are outside the scope of the thesis and more information can be obtained here [71, 72]. In summary, it implements a fixed time-step simulation synched to a real-time clock. The aerodynamic model was reused from a military simulation of the PC-9 (see Fig. 7.2). The aerodynamic model is a full six degree of freedom, non-linear model, with provision for modelling stall, undercarriage, and flaps.

FSIM is a flight simulator developed by Curtin University. FSIM can be flown by a joystick or mouse and implements a flight dynamic model of a current RAAF training aircraft developed at DSTO. The flight simulator provides a three dimensional out-of-cockpit view and a generic set of instruments. Atmospheric effects such as wind and turbulence are modelled. Input files specify geographic features such as airstrips, mountains, buildings and trees.

7.3.2.3 Instructor Agent Reasoning

The instructor agent was implemented in the dMARS programming language [42]. dMARS is a C++ reimplementation and extension of PRS [179]. It is a subset of Rao and Georgeff's BDI model and is well suited to modelling procedural reasoning of the type performed by the instructor agent. BDI languages have been utilised in military simulation and in domains such as process control, health monitoring and diagnosis, and air traffic control scheduling.

Figure 7.21: A screen capture from the virtual air show in flight. The graphical aspects of the interface development were reused from previously developed systems. The cockpit layout his a generic instrument panel that is not intended to portray any particular aircraft.

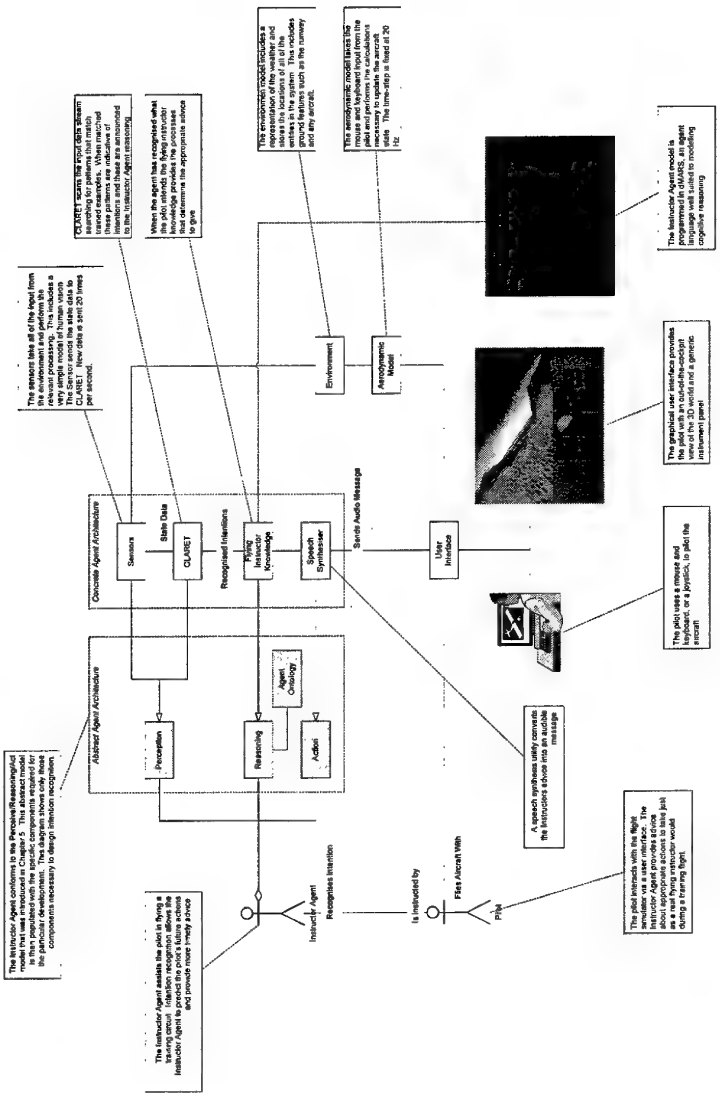

Figure 7.22: The VAS system architecture. An existing flight simulation system is to be augmented by the addition of an instructor agent. By correctly recognising the intention of the pilot the agent can better advise the pilot. The addition of agents to legacy systems that interact with humans is perhaps the most obvious and common class of agent systems.

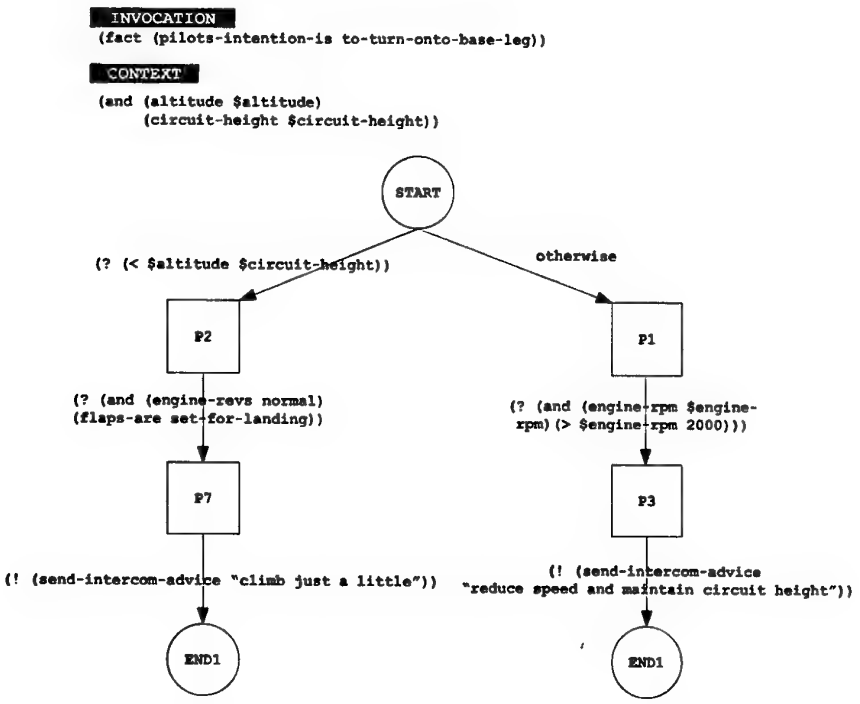

Figure 7.23: An example dMARS plan from the Instructor Agent. These plans provide 'recipes' that determine the means to the goals, or 'ends'. Plans in turn can invoke other plans leading to nested structures of instantiated but partially executed plans. These structures are the agents 'intentions'. This is consistent with the notion of intentions as a commitment to a plan to achieve a goal.

A suite of plans [146] was developed that give it the capacity to assist the pilot. The knowledge required by the Instructor Agent about how to fly a circuit and the advice to provide to the pilot is specified in terms of these plans. The execution semantics of the BDI model and dMARS is not relevant but can be referenced [149]. Plans in the dMARS system are graphical representations of the procedural knowledge required to complete a task [146]. These plans rely on external information for execution. These data take two forms. (1) Data from the flight simulator. Typical examples are altitude, heading, roll angle, pitch angle, engine revs. These data are typical of the types of data that a pilot would normally acquire from the flight instruments. (2) Data from the machine learning module. Typical examples are; "finished base-leg", "finished take-off roll", and "engine failure". These data are examples of the types of data that a pilot infers from examining his environment and complex relationships between entities. As these plans execute, they follow the progress of the pilot. The agent-instructor issues messages to the pilot guiding him through the circuit and advising about the manoeuvres to employ.

7.3.2.4 Instructor Agent Perception

The environment will provide a time-stepped feed of variables that indicate the aircraft positions, velocities, angles and angular rates, as well as the control positions, switches, and settings that are adjusted by the pilot. Together these provide the raw data that is available to the instructor agent in recognising the intention of the pilot. These are pre-processed to provide the data required by the perception module. The design as described requires that the perception module must map these spatio-temporal time histories directly into symbolic descriptions of the intentions as they are recognised.

A pattern matching system, CLARET, was chosen as a suitable means of performing the sophisticated mapping required. A brief description of CLARET is included here for completeness though the detail is outside the scope of the thesis. Further information can be obtained here [137] and specifically with respect to recognising aircraft manoeuvring here [139].

Machine learning when applied to handwriting recognition has proved very successful at recognising spatial trajectories in two dimensional space. Once it is trained, CLARET, an algorithm based on relational evidence theory, incorporates temporal aspects in order to prune the search space dynamically and hence successfully predict the particular alphanumeric symbol before it is completed.

In descriptive recognition, an expert pilot explains the relationships between event types specifying a decomposition of high-level manoeuvres - "tell me about these manoeuvres". For example, flying circuits can be decomposed into take-off, crosswind, down-wind, base-leg and final-approach manoeuvres. Pattern matching is then used to bind these manoeuvres to traces of simulator activity. The system dynamically binds to different intentions as they manifest as trajectories of input time series. The technique used is based on statistical pattern matching and learning techniques, currently used for on-line handwriting and gesture recognition. In the CLARET algorithm an unknown, segmented, and labelled trajectory case is presented to the system together with examples of known trajectories using a simple polygonal approximation technique. First, relationships between trajectory segments are extracted and their relationships calculated. Relational rules are

generated that explicitly depict relationships between states. For example, a right-turn manoeuvre is defined by a subsequence of different roll-pitch-yaw states (see Figure 7.24) over time.

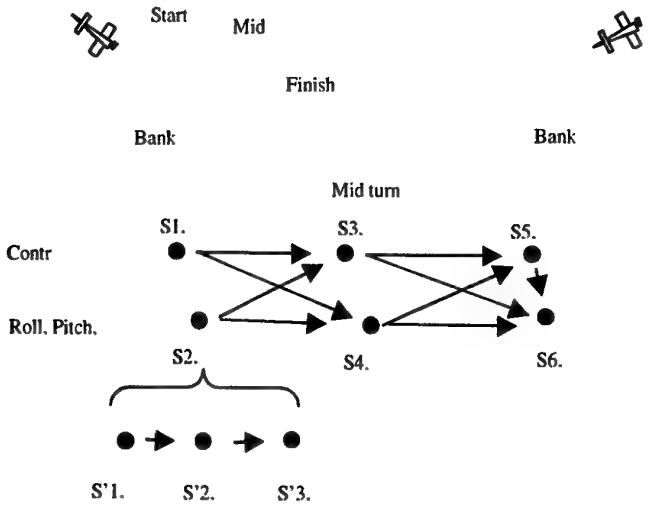

Figure 7.24: Partitioning a Manoeuvre

Second, matching techniques relate intentional descriptions to the control actions and available information from the displays (instruments and external 'out-the-window' world-view). A parser is then used for conversion of events matched using these rules into descriptions that are consistent with the syntax of the events and their relationships. For example, the turn-to-base leg intention is defined by the relationships of different events relative to wind direction, heading and runway position:

```
While in context of FLYING-CIRCUIT
  If LEFT-TURN at CRUISING-ALTITUDE and
  If LEFT-TURN overlaps WIND-AT-RIGHT-ANGLES
Then intention is TURN-TO-BASE-LEG
```

This is human-readable output can be used to interactively validate partially enacted sequences in the simulator. The system uses inexact (approximate) matching and a Bayesian probability network to negotiate between alternative hypothesis, and thus the rules are transferable to other airports and under variable conditions (for details of the CLARET algorithm see [137].)

Thus referring again to the example of turning base-leg the pattern matching algorithm observes is continually supplied with aircraft state information and pilot control information. These spatio-temporal trajectories are matched against a library of pre-trained examples and when a match is obtained, (or rather when the probability of a match exceeds some threshold) a match is announced to the dMARS agent in the form of a direct belief assertion into the agent's database. This assertion triggers a reasoning process based upon the recognition of the intent. In the case of the turning-base-leg example it turns out that the geometric relationship of the aircraft to the runway is the most

significant data in characterising this particular intention[61].

7.3.3 Real Time Intention Recognition by Pattern Matching

7.3.3.1 System Integration and Testing

With well defined interfaces and mature technologies for many of the components the system proved to be relatively bug-free, easy to test and system integration was reasonably straight-forward.

The use of a pattern matching algorithm as the primary component of the perception module simplifies the software development somewhat because the rules governing the mapping from system data to agent knowledge are trained not coded.

Once the system was integrated and primary testing complete the task of training the system to recognise the appropriate intentions was undertaken.

7.3.3.2 Training

The training process begins by deciding upon a categorisation, or ontology, of the examples to be learned. This ontology then forms the basis for the training with examples of each of the elements provided. This ontology is provided to the VAS in the form of a structured data-file that is read-in as a part of intialisation. Thus the training of the recognition software is matched exactly to the ontology that defines the intentions to be recognised.

A keyboard command places the VAS into a training mode. In this mode the user selects one of the ontology elements from a displayed list and then proceeds to fly an example of that. The examples chosen need not be manoeuvres as such, although manoeuvring may be a big part. As the aircraft is flown CLARET records data from the simulation and stores it as an example of that particular element in the ontology.

Training proceeds until the elements of the intention recognition ontology have been covered. Increasing the number of examples of each of the patterns improves CLARET's chances of matching if the expected variation between members of the same pattern are expected to be high in practice but also increases the time taken to announce a match. If the examples of each pattern encountered in operation are quite similar and if there are large differences between the categories then there is less need for more examples. Increasing the number of examples results in an exponential computational performance penalty.

In practice between five and ten examples of each pattern were enough. Performance did not become an issue until the number of examples rose to approximately forty indicating a reasonable amount of computational headroom. The interface between the pattern matcher and the simulation systems was configurable to allow user selection of the data

[61]Though the position of the aircraft to the runway seems likely to be very important in recognising all of the intentions associated with circuit flight experimentation showed that CLARET tended to classify based on altitude or speed with runway position being secondary.

that was input to the pattern matcher [62]. An important part of the commissioning process were the decisions concerning the supply of data to the pattern-matcher. This is discussed in greater detail later.

CLARET selects the set of input data those that disambiguate the different patterns. There is again a performance penalty for increasing the attributes that CLARET uses to differentiate patterns. For this reason it is preferable to use a small useful subset of the attributes available. Testing of the system provided insights into pattern matching process.

Speed is a very important concept, perhaps a very close second to altitude, in a pilots decision making. Speed, however, proved to be not such a useful classifier of the different phases of flight. This is best illustrated by Fig. 7.25 that shows a typical trace of altitude versus time for each of the phases of a circuit. Take-off and landing are clearly identifiable but other phases of the circuit are very similar.

A concept that did prove to be useful in classifying the pilot's intention was "range to runway" (See Fig. 7.25). From the typical graphs it can be seen that this clearly distinguishes each of the circuit phases. "Range to runway" is not a concept that is immediately obvious in selecting the appropriate input data. *Range-to-runway* is a constructed value, being neither a state of the aircraft or of any other component in the simulation but rather a relationship between two things. It was added to the system ontology, the design, and the interface during this phase of testing when it was discovered during testing that it was a good classifying attribute. This is yet another example of how the environment might be engineered to assist agent perception. Though this example escaped the design process and was not discovered until system commissioning.

7.3.3.3 Operation

Once provided with the set of training examples the system is ready for operational use. While the simulation is in progress CLARET continuously searches the input data stream for recognisable patterns. If a pattern is matched to the exclusion of other possibilities then CLARET will report that recognition has been achieved. This is the standard functioning mode of the system.

Greater functionality is also available. CLARET is capable of serving queries from the agent regarding the current status of the recognition process. If two or more possible candidates remain in the set that CLARET is attempting to match then they can all be reported together with a indication of their relative likelihood. If an intention is recognized and reported then CLARET can serve queries about the nature of that manoeuvre relative to its set of trained examples. In this way CLARET can provide extrapolative prediction of expected future behaviour and add detail to the nature of the recognised intention.

Though not tested in this implementation CLARET, with a flat ontology, CLARET is capable of recognising hierarchically nested patterns and hence classify intentions at various levels of abstraction as long as it has been provided the training examples. This means that a more structured ontology can be matched at all levels of the hierarchy.

[62] The data is required to be identical between the training sets and operational runs.

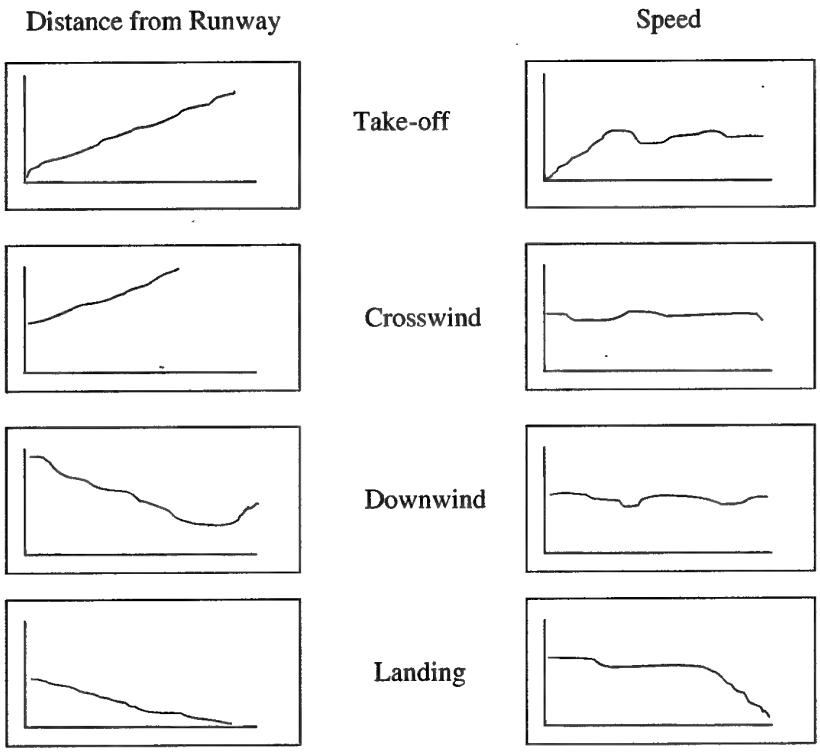

Figure 7.25: Sample traces from the trajectories from a circuit flight. The distance from the runway is a better attribute for categorising the intentions than speed. Though take-off is clearly distinguished by speed the other three phases of the circuit are very similar. CLARET uses many attributes to categorise and classify patterns but a computational overhead results from selecting a greater number so some care should be taken when determining appropriate attributes. By analysing the problem and supplying CLARET with appropriate constructed variables pattern matching can be improved. Further details are beyond the scope of this thesis but it is yet another example of how the environment might be engineered to support agent perception.

7.3.3.4 Performance

Processing is delayed to synch to the clock but the time step is not variable so processor delay beyond real-time causes *slowing* rather than *jumping*. The graphics subsystem is updated synchronously and therefore suffers from the same potential problems with real-time lag. Fortunately, the system is fast compared with the available hardware with more than 400% headroom. The available computational time was partly used by the perception subsystem that was also implemented synchronously in the same process. The agent, however, executes on a separate process and as the execution platform was multi-processor did not impinge greatly upon the real-time capability. At some of the highly loaded moments the system did slow to less than real-time, but never for more than a few time-slices.

7.3.4 Results and Discussion

CLARET was trained to recognise eight different intentions associated with circuit flight. The number of training examples was limited to five. Three different pilots were used to train CLARET and the same three flew the aircraft under the control of the Instructor-Agent. The pilot who performed the training was not always the pilot who flew the circuit. Every possible permutation of trainer and pilot was attempted.

CLARET successfully identified the eight flight conditions without error during every instance of accurate circuit flying. CLARET passed this information concerning the recognition process to the Instructor-agent. The agent used this information to assist in the modelling of the pilot's mental state. The agent successfully guided all of the pilots around the circuit by providing real time advice about actions to take. Because the pilot needs some warning about when and how to perform a manoeuvre it was essential that the instructor-agent provide the advice in a timely fashion. In every case CLARET was able to recognise the manoeuvres and the agent was able to process the data well before the information was actually required. If the circuit was flown poorly then CLARET was unable to match to the trained examples. The errors necessary to cause CLARET to misidentify flight conditions were such that a conscious effort needed to be made to fly poorly. During normal flight even by inexperienced users the systems functioned without error.

7.3.4.1 Performance of CLARET

The success of CLARET, whilst important to the success of the demonstration, was secondary to the more general aim of testing the demonstrating the methodology computational feasibility of connecting a model of perception and a model of rule/knowledge-based behaviour within the constraints of a real time system.

The intentions that CLARET was trained to recognise were sufficiently different that the task of discriminating the phases of flight was not difficult when compared with other applications to which CLARET has been put [138]. The real success of the experiment was in demonstrating the potential of pattern matching systems such as CLARET to model perception in a manner that allows direct recognition of intention. In so far as the

objectives of this thesis were concerned the implementation of the *Ecological Recogniser* pattern succeeded in proving the feasibility of modelling intention recognition in a rich real-time intelligent agent environment using an explicit model of perception implemented with a pattern matching system that connects to a higher-level more abstract model of reasoning. Further details about these experiments are available [72, 71].

7.3.4.2 Discovery of Center of Visual Flow

The development of Gibson's theory of ecological visual perception were influenced by his experience in training pilot to land during the second world war. After some time he noticed that a the *center of visual flow* was an important feature part of learning to land an aircraft. As someone moves in the environment those things on the periphery of vision (assuming that they look in the direction that they are travelling) appear to move faster than those things that we are heading directly towards. When landing an aircraft the approach is proceeding correctly if the point on the runway where the touch-down is to occur is centered and stationary in the field of view. By explaining this phenomena to pilots they can more quickly be taught to land.

A pilot's perception tends to focus then an the end of the runway and if there is relative movement of the point then adjustments are necessary. When CLARET was presented with many parameters to discriminate the pilot intentions the ones that it *preferred* where those that were related to the relative angle between the aircraft and the end of the runway. It seems that CLARET was able to best discriminate pilot manoeuvres when the variable it was presented with related to some representation of the pilot's perception. This phenomenon requires further investigation to ascertain its validity. It might be that it was due only to the particular dynamics of the VAS.

7.3.4.3 Assisted Ecological Recogniser

Preliminary experiments were performed with an implementation of the Assisted Ecological Recogniser. This system utilised a connection between CLARET and the mouse and keyboard as a representation of the pilot's *actions*. These preliminary experiments reveal that it is more difficult to recognise intention based on examination of the aircraft control actions. Intuitively it would seem like a more difficult proposition to recognise intention based on the aircraft controls because it is ultimately the pilot's intention to determine the aircraft trajectory and not to execute some movement of the controls.

Chapter 8

Conclusions

"I may not have gone where I intended to go, but I think I have ended up where I intended to be."—Douglas Adams

The conclusions of this thesis are presented in four sections. The first, Section 8.1, summarises the primary contributions of the thesis as they were presented in Chapters 5, 6 and 7 and the secondary contributions as they were presented in Chapters 3 and 4. Section 8.2 reexamines the influences and motivations that initiated this research, considering the impact of the research findings on agent design in general and military simulation in particular. Section 8.3 examines limitations and of the research and areas that might be addressed in the future.

8.1 Summary

In addressing the stated aim of this research *"to provide the designer of agent systems with a practical approach to modelling intention recognition for a range of intelligent agent systems"* this thesis developed a view of intention recognition geared toward architectural design patterns and influenced heavily by the state of the practice of agent applications and the ecological psychology literature.

The consensus of the literature is that perception is a fundamental property of agency. Furthermore mainstream psychological theories of perception have a structural similarity with existing agent-system implementations of intention recognition prompting a fresh look at perception.

Agents, particularly intelligent agents, are assumed to be software entities that should manipulate more abstract concepts than those that characterise other types of software resulting in an inevitable requirement to 'bridge the abstraction gap' that results between agents and other components in the system. Perception was introduced as the means by which this abstraction gap is bridged. The situated nature of agents inspired a look at perception that focussed on two seemingly diverse but actually closely related areas: the theory of ecological psychology; and state-of-the-practice of adding agents to environments. Together these two seemingly unrelated topics provided insights into the design of intention recognition for agent systems.

A consequence of the model of perception adopted is an explicit focus on the concepts of importance to the agent. This is suggestive of the general field of agent ontologies, a view reinforced when reflecting on the close links that exists between communication and intention recognition, and communication and ontologies. Perception, from an ontology design perspective, becomes the means by which the *system ontology* is converted into the form required by the *agent ontology*.

A model of intention recognition was presented that described intentional behaviour and its corresponding recognition at three levels: the intentional level; the activity level; and the state level. This three tiered description is consistent with other related approaches to decomposing behaviour. When this model of intention recognition is combined with the lessons derived from modelling perception a set of six architectures for modelling intention recognition were derived.

These architectures were presented in the form of the mainstream patterns literature and the properties, features, advantages, and liabilities of each are described. These patterns were used as the basis for designing variations of a flight simulator, the Virtual Air Show. These variants provide examples of the implemented patterns and demonstrate the attributes by which the patterns are judged.

Recalling the introduction to this thesis, there are two primary contributions of this thesis that correspond to the two constituent parts of the stated aim:

- a set of six design patterns to support the designer of intelligent agent systems that require an intention recognition capability. These patterns are presented in the style of the mainstream software engineering literature [18] but are appropriately inspired by a consideration of the psychology literature; and
- a description of an implemented system that illustrates the application and utility of these design patterns and provides the basis for a critical appraisal.

Two secondary contributions were a by-product of the particular methodology adopted. In seeking inspiration for the design patterns the agents literature, software engineering literature and the psychology literature were examined. This gave rise to the two secondary contributions of this research introduced first in Section 1.2:

- an account of the importance of perception in modelling agent systems. In particular, the importance of an explicit model of perception as the means by which structures or concepts in the agent's environment are converted into representations that are appropriate for the agent; and
- a description of the manner in which ontologies assist intelligent agent system design and the relationship between perception and ontologies. Influenced by the developed model of perception an account of ontologies is presented that describes a means of integrating agent ontology design into mainstream software engineering. The ontology is seen as a product of the design or, if it is preexisting then a constraint over the design.

These are summarised in the following sections.

8.1.1 Patterns for Intention Recognition

Chapter 6 documents six agent system architectural patterns using the template description suggested by Buschmann et. al. [18]. These patterns are based on an architecture derived from a consideration of the importance of agent perception combined with a three tiered model of intentional behaviour (See Figure 8.1). Pattern languages offer a vocabulary for design that allows the software engineer to express and document options in a succinct manner. Taken as a single set these six alternatives form the beginnings of a *pattern language* for talking about designing models of intention recognition. The patterns offer the agent system designer alternatives for modelling intention recognition and provide the information necessary to make informed design decisions about the relative appropriateness of the various options The description of the patterns indicate the advantages and disadvantages of each and allow the adoption of a design that is appropriate to the software requirements.

8.1.2 Appraisal of the Patterns

There are two elements to the meeting the second part of the aim of this thesis "demonstrate that when undertaking the design of intelligent agent systems this modelling approach offers software engineering advantages". One is to document the specific advantages that the individual patterns offer relative to each other. The strengths and weaknesses of each pattern was evaluated in Chapter 6 and then again with the aid of a worked example in Chapter 7. Greater experience with the wider application of these patterns is desirable. This experience aside the patterns as they are expressed are useful, in that they do offer genuine alternative across a range of requirements. A summary of the patterns indicating their strengths and weaknesses is shown in Figure 8.2 a more detailed table is contained in text at Figure 6.8.

The second, and more difficult, task is to demonstrate that these patterns represent a useful and worthwhile approach to modelling intention recognition in intelligent agent systems. This is achieved through the design of six variants of the same system—the Virtual Air Show. The discussion of the application of the designs provide a clear indication of the relative properties of the patterns from several viewpoints. The *Ecological Recogniser* pattern variant of this system was taken to implementation to explore the properties of an implemented system. This variant was chosen because it was scientifically the most interesting and at the same time most challenging from an engineering perspective. A screen shot from this system is shown in Figure 8.3.

8.1.3 An Explicit Model of Perception

Perception is commonly mentioned in definitions of agency and there is an emerging consensus in the agents community that it is a fundamental and defining concept. Another concept that is commonly used to define agents is 'situated'. These two concepts together with the realisation that there was a structural similarity between the 'sense-and-infer' theories of perception and existing implementations of intention recognition prompted a closer look at perception and led ultimately to the ecological psychology literature.

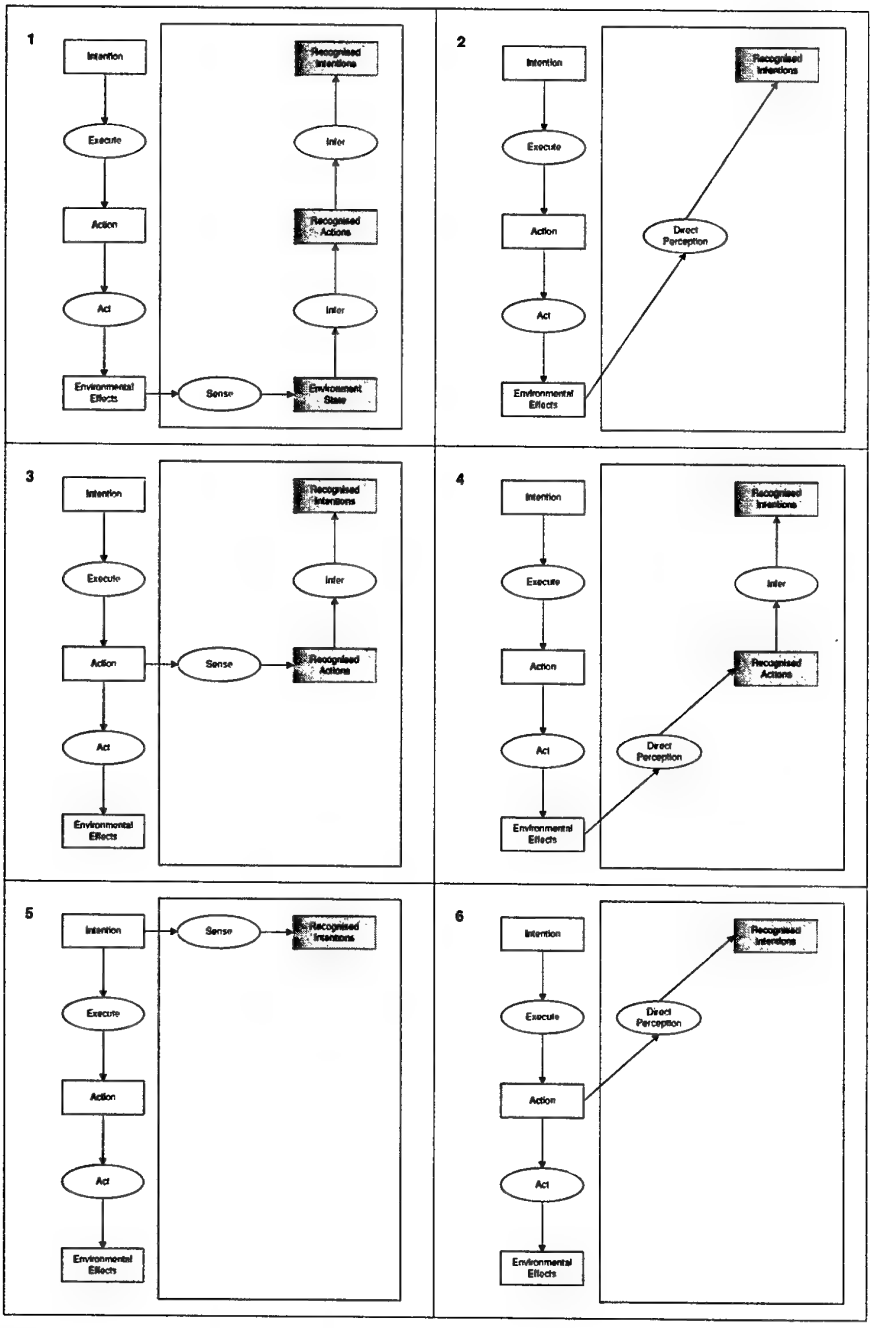

Figure 8.1: The agent architectural design patterns presented are variations on a theme developed by considering a model of perception inspired by situated agency overlaid on a three level description of intentional action. The result is a set of architectural possibilities that give rise to the six patterns presented in Chapter 6 provide a variety of characteristics that can be matched to the functional and performance requirements of a system.

	Hybrid	Sense and Infer	Assisted Sense and Infer	Ecological	Assisted Ecological	Clairvoyant
Description	A mix of the ecological pattern and the sense and infer pattern	This pattern is the one most commonly found in practice. Simply, the agent sense the environment and infers higher order properties.	A variation on the sense and infer pattern that links a representation of the *actions* of the *intending agent* directly to the *recognizing agent*. These representations of action are then processed by the cognitive reasoning of the *recognizing agent*.	A pattern where intention recognition occurs completely within the *perception* module of the *recognizing agent*. There are particular technologies that are indicated by this pattern.	A variation on the ecological pattern that links a representation of the *actions* of the *intending agent* directly to the *recognizing agent*. These representations of action are then processed by the perception of the *recognizing agent*.	A simple pattern where the *recognizing agent* is provided with a direct representation of the intent of the *intending agent*.
Suitable for Human	Human or agent	Capable of recognizing the intent of agents and humans but better suited system where the environment state is simpler (this suggests agents).	For some systems it might be possible to gain direct access to the human's actions. This is the case when those actions are easily labeled – for example the activation of some interface.	Capable of recognizing the intent of agents and humans but better suited system where the environment state is complex (this suggests humans).	For some systems it might be possible to gain direct access to the human's actions. This is the case when those actions are easily labeled – for example the activation of some interface.	A human version might be possible if provision is made for communicative interrogation by the agent about the nature of the human's intent.
Location of Intention Recognition	Perception and Cognition	Cognition	Other agent and Cognition	Perception	Other agent and Perception	Other agent
Primary Advantages	The best match of technology to problem of any of the patterns.	A relatively simple and common architecture that isolates the intention recognition processing inside the agent.	A sensible simplification of the Sense and Infer pattern that simplifies the inference to a level suitable for processing inside the agent.	Allows the agent's cognitive module to reason with the appropriate data.	Isolates cognitive and perceptual aspects of intention recognition in a straightforward manner.	Provides the simplest possible solutions. It is both architecturally simple and also likely to be technologically simple. Will almost certainly provide the best performance.
Primary Disadvantages	Technologically and architecturally the most complex pattern.	Adds complexity and low-level data to the agent reasoning module	Couples the agents by requiring that the intending agent provide direct access to its *actions*.	Requires a sophisticated pattern matcher or similar technology to extract the intentional patterns from the effects in the environment.	Doesn't make best use of ecological perception.	Lack of cognitive realism Lack of detail Agents are strongly couple More difficult to deal with human intention recognition Coupling

Figure 8.2: Summary of the characteristics of the six intention recognition patterns.

Figure 8.3: The Virtual Air Show was developed as a demonstrator of some of these ideas. This provides an example of the design impact of the various choices and the types of systems for which they are appropriate. The single implemented example shown here is an implementation of the ecological recogniser *pattern. The implemented system user a pattern matching system—CLARET—to provide the model of perception that feeds a high-level agent with direct representations of intent.*

Perception was described as a process within the agent that transforms data from the environment into the form required by the agent. Characterising it in this fashion suggests that it is a software interface. In standard software development, interfaces between system components must often convert data but intelligent agents typically operate at the 'knowledge level' and manipulate higher order concepts than those found in other varieties of software. Thus the interface between agent and environment must, in addition to other data transformations, abstract the data or otherwise transform it into knowledge level representations suitable for agent reasoning. Perception is a perfectly appropriate metaphor for this process in intelligent agent systems. Not only does it accurately reflect the nature of the interface but it carries the anthropomorphism often sought when describing intelligent agent concepts.

In the practical experiences of situating agents in environments came a second source of insight. Examples of labelled environments and labelled agents were presented that demonstrated how aspects of perception can be facilitated.

Together these two views of situated perception and agency, one highly theoretical and one highly practical combine to provide possible options for the implementation of perceptual behaviours in agent systems.

8.1.4 Ontologies and System Design

With an explicit model of perception providing the translation of data into the form required by the agent it is relatively straightforward to see perception as an ontology translator. Every software system, indeed every system, can be characterised, post-hoc, by an ontology. In some systems the ontology is more obvious than in others. In some it might be more clearly expressed in the design than it is in the executable system and in some it is barely visible at all. How seriously ontologies are teated during design depends ultimately on the importance of structuring and representing knowledge to the functioning of the system. Software that makes use of abstraction as a means of dealing with complexity is more likely to benefit from a careful consideration of knowledge structuring and hence require ontologies. If an ontology is pre-specified and mandated then it ought to be considered as a constraint over the agent-system design. If it is not pre-specified then it ought to be considered a product of the system design.

Two ontologies are useful for characterising agent systems. One that has the scope of the domain and expresses the global concepts that are useful across and between components and an agent ontology that has scope of the agent and expresses the concepts local to the agent. Perception mediates between the agent and the rest of the system by translating between these ontologies.

Ontologies can structure knowledge of intentions enabling the modelling of intentional action and the modelling of intention recognition. That these two processes are different advocates local ontologies and, if local ontologies are preferred, then a translation mechanism is required—perception fills that role.

8.2 Revisiting the Thesis Influences

As discussed in Sections 1.4 and 1.3 there are a number of motivating factors that have influenced this research. These are briefly recounted here.

8.2.1 Agent System Design

Several insights for the design of agent systems are contributed by this thesis. The following short summaries are examples of small contributions made by this thesis to the broad discipline of agent design. In most cases it is difficult to draw strong conclusions, but they are presented here as single data points in the broader study of agent design that might be examined in more detail as a part of future research or combined with other similar experiences to draw more general conclusions.

8.2.1.1 A research focus on real world problems

Much agent research is not focussed directly on real-world problems but becomes trapped in *toy-worlds* where it is difficult to learn lessons that translate into industrial and commercial settings. The types of military systems that initiated this research typically require between 10 and 20 person-years of development effort. This is clearly beyond the scope of postgraduate study which creates problems for evaluating research into system design. This thesis focussed on technology-independent design that removed the need for the time consuming overhead of large scale systems implementation. It is not clear that this would have been possible had there not been considerable prior experience with the development of very large agent systems [76].

8.2.1.2 Intentional analysis as a means of specifying agent behaviours

Intentional analysis is a necessity of a software system that must exhibit intention recognition. Elsewhere it has been suggested as a useful approach for analysing requirements in a range of information systems [59]. In Section 5.2.4 the idea of intentional analysis was introduced and then taken further with some practical examples of a possible application in Section 7.3.1. Further research is necessary to explore the idea of intentional analysis and to extend it into a mainstream requirements engineering tool.

8.2.1.3 An approach to documenting architectures of agent solutions

One of the clear impediments to the proliferation of agent technologies is a collection of well documented examples of successfully deployed agent applications. The experience of the object oriented community with design patterns has clearly indicated the benefits of design level reuse. So whether the issue is direct design reuse, or simply understanding the solutions that others have found useful, design patterns offer a standard template for expressing an agent system design that is useful and familiar to software engineers. This thesis presented design patterns as a useful way of documenting an agent system.

8.2.1.4 Designing agents as post hoc additions to legacy systems

For anything but *toy* systems agents are seldom developed in isolation. They will of necessity exist within and along side other forms of software. The systems that agents inhabit are usually, though not always, large legacy systems that must be retrofitted to suit the agent or must be dealt with, unmodified, by the agent. By expressing design patterns that offer implementation alternatives and by introducing an approach to ontology design that considers trade-offs between the environment and the agent this thesis explicitly caters for the design of agents in environment and offers techniques for designing environments for agents.

8.2.1.5 A step toward mainstreaming agent technologies

If agents are to enter the mainstream they will be supported by the collective experiences, methodologies, and practices that make up software engineering. This thesis has focussed not on agent technologies, tools, or languages, which are well represented in the literature but has chosen to focus on the other aspects of software engineering that are less well represented: design, architectures, and patterns.

8.2.1.6 Demystifying of the "black-art" of ontology design

By developing a model of the *agent ontology* as distinct from the *domain ontology* and then declaring perception to be the mechanism responsible for translating from the latter to the former it is relatively simple to derive an approach to software engineering that sees a consideration or design of the ontologies as something integrated within standard software engineering.

8.2.1.7 Toward a unified definition of agency

Definitions of agency vary and agreed definitions although unimportant for science are important in developing software engineering standards, methodologies and for pedagogy. In this thesis a definition of agency was provided that is consistent with much of the literature. This definition is high level and general but the detail is important to the elaboration of the models presented. In particular a definition of perception was presented that informs more general views of agent design.

8.2.1.8 Agents are knowledge level entities that often inhabit less abstract environments

The need to situate agents in environments prompted the focus on perception as the means by which the data in the environment is converted into the form required by the agent. In particular the idea of maintaining 'ontological purity' reflects a desire by some agent designers to avoid the use of low-level data inside the agent and demonstrates a belief that agents should be described at a different level of abstraction to other types of software. A desire to maintain the high level knowledge level reasoning aspects of the

agent as ontologically pure as possible thus maintaining the level-of-abstraction advantage agents have whilst recognising the need to interface with other types of software.

8.2.1.9 Solutions to designing intelligence will require hybrid technologies

The architectures developed specifically support modularisations that allow for hybrid technologies to be used for implementing intention recognition. The Hybrid Pattern(Section 6.2.1) and the two ecologically inspired patterns (Section 6.2.4 and 6.2.5) support hybrid technologies. The example implemented system in Section 7.2.4 provides an example and a discussion of a hybrid system. Some researchers consider it as fundamental to the future of artificial intelligence that architectures that combine disparate technologies are available. The approach presented in this thesis certainly does not preclude hybrid systems and some of the patterns implicitly encourage it.

8.2.2 Military Simulation

Until the recent flourishing of console-based games and the emergence of computer graphics in animation and special effects in movies, military simulation was the primary driver of many aspects of simulation technology. Applications for intention recognition are widespread but military simulation remains an important consumer of any technology developed in this field. Military simulation fostered this research with a stated requirement to develop the capacity to simulate intention recognition within the range of simulation applications. These include real-time human-in-the-loop simulators and faster than real-time constructive simulators used for operations research. Requirements often conflict and fidelity, plausibility, and cognitive realism must often be traded off against performance, simplicity, maintainability and explainability. Intention recognition is an identified need of the military simulation community but there is such variation in the specific requirements that a range of solutions will always be necessary.

A modelling framework and a set of architectures that offer alternative implementations of intention recognition. This allows the developers of a particular simulation to consider their particular requirements against the range of options presented and select an appropriate architecture. Simply developing a single technology, language, or architecture presupposes a set of requirements that will inevitably fail to cover the spectrum necessary for meeting the needs of the military simulation community.

Military simulation systems must be well engineered—they must be quality systems. This requires that they are well specified and well designed. Furthermore, non-functional requirements can dominate and performance, particularly in real-time simulation is often paramount. Inevitably there are compromises that must be made between what might be termed fidelity[63] and performance. Managing these trade-offs must be done explicitly, in order that they are documented and well understood. This thesis has presented a design approach that offers design models that allow these decisions to be made in a reasoned fashion.

[63]Fidelity here is a catchall term for credibility of model, psychological plausibility, verifiability, and the properties of the model that govern the manner in which it represents intention recognition.

From the implementation of the virtual air show described in Section 7.2.4 there are insights into a particular technology, the CLARET pattern matching system, and its application to the recognition of spatio-temporal trajectories. The particular requirements of military simulation often require processing of spatio-temporal data streams that indicate the trajectories of the simulated entities. Architectures and technologies for addressing these particular problems have been presented and discussed.

An approach to developing intention recognition that allows it to be integrated with other software engineering activities as *just another* required system functionality. The approach presented allows for ontology design, perception modelling, and intention recognition to be integrated into the agent system design process. A software engineering methodology has not been developed but the models presented in this thesis should not overly constrain any preference. Military systems typically require mode rigorous engineering than other systems and providing solutions that are compatible with requirements driven software engineering is critical to military simulation. Intention recognition is not seen as something special, just another behaviour that must be provided.

This thesis has strongly demonstrated the capacity of components of a system other than the agent to be modified to support the design requirements of the agent. The military simulation community has a large investment in existing simulation environments. Architectures that inform the design process that modify these environments to make them *agent friendly* are critical to extending the life of these simulators in the face of a need to incorporate agents in the form of computer generate forces.

Benefits from the direct application of this research are limited to those systems that conform to the assumptions of Section 5 and, more importantly, require intention recognition. Usefulness is dictated by whether or not the patterns characterise the intention recognition problem in a manner that allows the architectures to be reused over many application domains. Only experience with the patterns in several serious applications will test that result. Wider implications of this research are linked more to the approach adopted and the style of research conducted than to the specific results. The development of the particular military simulators that initiated this research is ongoing [76]. The need for an intention recognition capability in these simulations is increasingly urgent and there are already plans to implement systems based on the outcomes of this research. Specifically there are some important results of this for the designers of military simulation. These lessons apply to intention recognition and also to broader design issues.

8.3 Limits and Extensions of this Research

The thesis was introduced with the aim to "provide the designer of agent systems with a practical approach to modelling intention recognition for a range of intelligent agent systems".

This aim was successfully achieved—six agent design patterns have been presented that offer the agent system developer with many choices for modelling recognition but there are several areas that might be strengthened to advance this research further.

8.3.1 Experience with the Patterns

In order to be classified as patterns it is normal for an architectural design to have been used and reused several times. Although the architectures that have been developed throughout this thesis where presented in the style of the patterns literature greater experience with their application is necessary to strengthen the claim that they actually offer substantial benefits to the designer of agent systems. Some quantifiable evidence that design patterns actually reduce the development time for agent systems would strengthen the claims of software engineering advantage. There is strong evidence in the object oriented software development community that there are measurable advantages to be derived from adopting patterns. Obtaining this type of data in the world of agent oriented software development requires the existence of many large systems with well documented architectures. This thesis might be seen as a small part of the bootstrapping effort necessary to break a Catch-22: until software engineering practices for designing agent systems are available building large agent systems is difficult and until experiences with the design of large agent systems are published it is difficult to develop suitable design approaches.

8.3.2 Multi-Agent Intention Recognition

In multi-agent environments confusion as to the identity of the *intending agent* substantially complicates the intention recognition. Though outside the scope of this thesis many applications, notably military simulation, will require extensions to this research to cater for intention recognition in a socially complex setting. The general approach of this thesis could be extended to model multi-agent intention recognition. The extension would need to include a recognition of team structures and joint intentions and the ability to reconcile ambiguity in identity as well as activity. The modelling approach likely to be taken would see the introduction of a fourth level in the model of intentional behaviour. This *social layer* would indicate the structured social relationships existing between agents and that causally give rise to intentional behaviour. Reconciling identity could then be undertaken at the highest level as a prelude to recognising intention. This is a simplification of the approach suggested by Tidhar and Sonenberg [162]. A more complex solution would treat identity as orthogonal to intentional behaviour and adopt something of the same process as that indicated for intention recognition but acting on the social structures. A still more complex solution would treat social structures and intentional behaviour as mutually dependent and mutually causal. This creates an n by m hypothesis space in unknown social structure and unknown intention. It is likely that the combination of the work on organisation oriented programming by Tidhar [172] together with the related work on intention recognition would provide the most suitable starting point for developing the models upon which a set of architectural design patterns would be based.

8.3.3 Software Life-Cycle Support

This aim of this thesis was to "support the designer" and so stayed mute on the various other issues related to the software development life-cycle. Although examples of implementation technologies were given, and analysis was briefly touched upon, languages,

testing, validation, requirements specification, and the integration of intention recognition with other agent functional behaviours has been ignored. Extensions to this thesis might focus on particular implementation techniques, testing, or on other important facets of software engineering.

8.3.4 Agent Patterns

Of most interest to the agent oriented software engineering community is the extension of agent patterns into other areas of agent design. Patterns for multi-agent collaboration, internal agent architectures and extracting patterns from successfully deployed agent systems as they emerge are just three of the areas that deserve exploration. In the early days of any technology failures often outnumber successes and documenting those is valuable and insightful. Agent *anti-patterns* would examine implementations that have been shown to be unsuccessful [14]. Concentrating on the detail of intention recognition would allow for greater insights to be gained into the application and implementation of these patterns in industrial settings. It is to be expected that the results of this thesis will be applied almost immediately in ongoing research and development into military simulation. Beyond adding detail to the presented patterns, or extending the scope of agent patterns further there are several research areas suggested by this thesis.

8.3.5 The General Applicability of Intentional Analysis

Intention oriented analysis, briefly mentioned in Chapter 5 offers a new approach to analysing complex software systems. It allows the developer of a system to analyse a system by making use of the folk-psychology that supports their everyday activities and as such is a powerful and familiar tool. An intentional analysis is a prerequisite for a designing a system that exhibits intention recognition. Whatever constitutes an intentional analysis it is clear that it must result in some description of the system in terms of the intentions of the agents that populate it. There is some evidence that an intentional analysis has wider applicability than the self-evident case of supporting intention recognition. The argument posed here is based on Dennett's intentional stance.

Dennett offers three stances that can be adopted to predict the behaviour of complex systems: the physical stance; design stance; and the intentional stance [39]. Each stance provides explanatory and predictive power and is progressively more abstract than the last.

Some software systems, perhaps *even* some agent systems are capable of being understood from physical stance[64]. Slightly more sophisticated agent systems might be simple enough to be to be understood from the design stance. An understanding of the agent design is enough to reliably predict, and hence understand, the agent behaviour.

[64]This will be interpreted as meaning that the system can be understood by inspection of the source code. Perhaps Dennett might prefer machine code, or even the physical operation of the hardware as the machine code executes, but source code suffices to make the analogy.

More complex systems are not understandable from the design stance. An understanding of the design of the agent is inadequate for predicting its behaviour[65]. Complex agent systems, particularly those that inhabit dynamic environments that expose them to continually changing, unforeseeable, combinations of input react according to their internal design, a design that is often either necessarily complex (in the case of symbolic reasoning systems) or hopelessly incomprehensible in the case of neural or subsumption architectures. But put these systems into the environment and their behaviours appear sensible, practical, intelligent, and above all understandable.

Dennett would claim that in this case the explanations of agent behaviour come from one of two sources: either a deep and thorough understanding of the software design and the state of the system, or a resort to the intentional stance—a folk-psychological view of agents that is extremely reliable in its predictive capacity.

Some agent systems are simple and can be understood from the design stance. For the more complex ones, and those with high-level behaviours like intention recognition are obvious candidates, the intentional stance provides a more abstract view of the system that is at once easy to use due to its familiarity, and yet still useful in its reliability and software engineering utility.

This is not to discount software design. Clearly all software must be designed and its behaviour clearly and unambiguously specified. Intentional descriptions of system behaviour can be useful through the software life-cycle and can certainly provide input into the design process but they are clearly not a substitute for design[66]. This then is the basis of the case for the universal applicability of intentional analysis to complex agent systems.

8.3.6 The AI Debate about Perception and Cognition

Drawing the line between perception and cognition has led to some of the more heated debates in AI [26]. In a different though superficially related way much of the connectionist/symbolicist debate has moved from claims by either side to outright dominance to broad agreement that both approaches have merit. There are clearly some tasks to which symbol systems are better suited and some to which network based approaches have the upper hand. Deciding where to prefer one style of solution over another impacts upon many aspects of AI including where the line between perception and cognition is drawn.

The impact of the ecological psychologists on this debate has been minor and the debate is about where to draw the demarcation line inside the agent. There are relatively

[65]The rather frightening proposition is that in complex software systems the system design is an inappropriate means of understanding the behaviour of the software. Neural networks are an interesting example of this phenomenon. Predicting the behaviour of a neural network when exposed to a input might be possible from knowledge of its internal design and its state but it is simpler to talk in terms of the net being 'trained' and being capable of 'recognising'. These concepts are abstract and though they characterise the function of the neural network they are not an intrinsic part of the design. Even in simpler agents the dynamism and richness of the environment can lead to complex unpredictable behaviours from simple agent designs.

[66]The one possible exception to this would be agent systems that are designed and programmed using explicit representations of intention. Though no such languages exist the logical framework has been developed [148].

few proposals from AI researchers to include the aspects of the system outside the agent to support perception (or possibly even cognition).

This thesis has presented a set of architectures that ignores conventional ideas about what is cognitive and what is perceptual and distributes functionality on the basis of software engineering pragmatics. Even so each of the patterns is justifiable in a cognitive modelling sense and as a whole they serve to map the landscape of possibilities and may give rise to approaches that have been previously undervalued or ignored.

Care must be taken in drawing the line between perception and cognition. The six patterns developed each model the division between perception and cognition (agent reasoning) in different ways. Rather than take sides in one of AI's ongoing debates this thesis provides support for modelling the perception cognition demarcation in many ways.

Perception and sub-symbolic aspects of cognition are often ignored by AI researchers. By providing an explicit model of perception as a part of the characteristic model of agency this thesis forces a consideration of agent perception. The encapsulation supported by the model of perception allows agent developers to ignore the perception related aspects of the agent whilst they develop the agent reasoning if they so choose. The perception module must eventually be designed but the compartmentalisation is a useful one.

8.3.7 Use Cases for Documenting Intentions

Use cases are a technique from Object Oriented Software Engineering (OOSE) [82] that has been adopted by the mainstream object oriented community and is supported by the UML and many case tools. Use cases were introduced to enable the specification of requirements in systems where the interactions between the user and the system are complex.

That a use case analysis commences by considering scenarios sets the it apart from other OO techniques. Scenarios allow the future user of the system to visualise the expected interactions and to describe the system in terms of the activities that it must support and the functionality that it must provide. Typically a scenario of the system in use is analysed and system functionality is specified and structured by the use cases. Heinze and Papasimeon have shown that a Use Case analysis can be successfully adapted for specifying behavioural requirements in agent systems [73]. Use cases equally be used to document and structure, admittedly in an informal way, the intentional behaviour of agents. Activity and interaction diagrams can indicate temporal aspects of the intention. In the example of Chapter 7 use cases documented the intentions of the actors involved. Use cases can show interactions between agents, human users, and other components of the system. Further research might examine the capacity for use cases to form the basis of a language for modelling intentions so that intention recognition can be analysed and designed.

8.3.8 Recognition versus Anticipation

There is a class of intention recognition that does not explicitly require an opponent. For example Laird [103] has constructed a *Quakebot*[67] that builds a mental model of an opponent in order to anticipate them. An interesting feature of this approach is that there is no requirement for any observation of the opponent to exist for its actions to be predicted. It is assumed that an opponent exists and that they can be anticipated via an empathic simulation process. This does not qualify directly as intention recognition per se. It is intention prediction. Even so, there are a number of modelling options that might support this application that fit within the design patterns presented. Rather than focussing on the *intending agent* a recognising agent could perceive the opportunities offered by the environment for intentional behaviour. This information would support hypotheses allowing the anticipation of intent.

Epilogue

When building software capable of human-like behaviours, there is a temptation to be drawn back into *good old-fashioned AI* (GOFAI) and the quest for computational representations of our brain that are all-powerful, elegant, and in some sense representative of the true state of our physiology. But if practical software development is the aim, and the software is resource bounded (as it always is) then there are inevitable trade-offs between the ambition of GOFAI and software engineering pragmatism. Whilst cognitive scientists search for ever better representations, the engineer has to build systems today, and these systems (perhaps inspired by cognitive science) must ultimately meet the requirements that rightly drive their development. The short-circuits, tricks, and *unnatural* designs are justified precisely because they do not match any reality. Besides, as Dan Dennett says—*"the brain cheats!"*—and if the brain cheats, why shouldn't the software engineer?

References

1. Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Schlomo Angel. *A pattern language: towns, buildings, construction.* Oxford University Press, New York, NY, 1977.

2. Gertrude Elizabeth Anscombe. *Intention.* Harvard University Press, 2000.

3. M. Barès, D. Canemero, J. F. Delannoy, and Y. Kodratoff. Xplans: Case-based reasoning for plan recognition. *Applied Artificial Intelligence*, 8(1):617–643, 1994.

4. Bernhard Bauer. Extending UML for the specification of agent interaction protcols. Response to the OMG Analysis and Design Task Force ad/99-12-03, Object Management Group, December 1999.

[67] A Quakebot is an artificial opponent in the computer game Quake. Quake provide a programming interface that allows for the construction of these.

5. V. Benjamins, D. Fensel, and A. Prez. Knowledge management through ontologies. In *Proceedings of PAKM-98*, pages 5.1–5.12, 1998.

6. Federico Bergenti and Agostino Poggi. Exploiting UML in the design of multi-agent systems. *Lecture Notes in Computer Science*, 1972:106–121, 2001.

7. Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide.* Addison Wesley, 1998.

8. Marco Bozzano, Giorgio Delzanno, Maurizio Martelli, Viviana Mascardi, and Floriano Zini. Multi-agent systems development as a software engineering enterprise. In Gopal Gupta, editor, *Practical Aspects of Declarative Languages*, number 1551 in Lecture Notes in Computer Science, pages 46–60, 1999.

9. Jeffrey Bradshaw. *Software Agents.* MIT Press, 1997.

10. Michael Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, 1987.

11. Michael E. Bratman, David Israel, and Martha Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

12. Frances M. T. Brazier, Barbara Dunin-Keplicz, Jan Treur, and Rineke Verbrugge. Modelling internal dynamic behaviour of BDI agents. In *ModelAge Workshop*, pages 36–56, 1997.

13. R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

14. William H. Brown, Raphael C. Malveau, Hays W. McCormick III, and Thomas J. Mowbray. *Anti Patterns.* Wiley, 1998.

15. V. Bruce and P. Green. *Visual Perception: Physiology, Psychology and Ecology.* Lawrence Erlbaum and Associates, Hove and London, UK, second edition, 1990.

16. J. C. Brustoloni. Autonomous agents: Characterization and requirements. Computer Science Technical Report 91-204, Carnegie Mellon University, 1991.

17. B. Burmeister. Models and methodology for agent-oriented analysis and design. In K. Fischer, editor, *Working Notes of the KI'96 Workshop on AgentOriented Programming and Distributed Systems*, Germany, 1996.

18. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture*, volume 1 of *Wiley Series in Software Design Patterns.* Wiley, New York, NY, 1 edition, 2002.

19. P. Busetta. Discussion paper on the dMARS-R recognition kernel proposal and prototype. Contractor report, Air Operations Division, DSTO, Melbourne, Australia, 1997.

20. Paolo Busetta, Ralph Ronnquist, Andrew Hodgson, and Andrew Lucas. Modelling and design of multi-agent systems. *AgentLink News Letter*, January 1999.

21. Chris Butcher and Jaime Griesemer. The illusion of intelligence: The integration of ai and level design in Halo. In *In Proceedings of the Game Developer Conference*, San Jose, California, March 2002.

22. by J. M. Rolfe and K. J. Staples. *Flight Simulation.* Cambridge University Press, 1988.

23. C. and G. Merriam Co. *Webster's revised unabridged dictionary. Online at http://www.dictionary.com/*, 2003.

24. J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: The tropos project. In *Information Systems.* Elsevier, 2002.

25. Richard Catrambone, John Stasko, and Jun Xiao. Anthropomorphic agents as a user interface paradigm: Experimental findings and a framework for research. In *Proceedings of CogSci 2002*, 2002.

26. D. J. Chalmers, R. M. French, and D. R. Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4(17):185211, 1992.

27. E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.

28. Liren Chen and Katia Sycara. WebMate: A personal agent for browsing and searching. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York, 9–13, 1998. ACM Press.

29. Paolo Ciancarini and Michel Wooldridge. *Agent Oriented Software Engineering.* Springer, 2000.

30. William Clancey. *Situated Cognition.* Cambridge University Press, 1997.

31. Mark L. Cohen and David Chong. *Microsoft Flight Simulator 2000.* Prima Publishing, 2000.

32. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

33. A. Collinot and A. Drogoul. Using the cassiopeia method to design a soccer robot team. *Applied Artificial Intelligence*, 1997.

34. Nuno Meira Conde. A set of agent patterns for a more expressive approach. In *Proceedings of the EuroPLOP'2000*, 2000.

35. R. Scott Cost, Ian Soboroff, Jeegar Lakhani, Tim Finin, Ethan Miller, and Charles Nicholas. Agent development support for Tcl. In USENIX, editor, *5th Annual Tcl/Tk Workshop '97, July 14–17, 1997. Boston, MA*, pages 177–178, Berkeley, CA, USA, 1997. USENIX.

36. S. Cranefield and M. Purvis. Uml as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

37. J. Curmi, R. Fiusco, R. Keeling, K. King, S. Pickup, G. Tidhar, and G. Tudor. Modelling pilot behaviour in air traffic control simulation. Technical Note 70, Australian AI Institute, Melbourne, Australia, November 1997.

38. S. DeLoach. Multiagent systems engineering a methodology and language for designing agent systems. In *Proceedings of the Workshop on Agent Oriented Information Systems*, 1999.

39. Daniel Dennett. *The Intetional Stance.* MIT Press, 1989.

40. D. Deugo, M. Weiss, and A. Case. A case for mobile agent patterns. In *In Proceedings of the Workshop on Mobile Agents in the Context of Competition and Cooperation at Autonomous Agents '99*, pages 19–22, 1999.

41. Dwight Deugo, Michael Weiss, and Elizabeth Kendall. Reusable patterns for agent coordination. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies and Applicaitons*, chapter 14. Springer, 2001.

42. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dmars. In *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages*, number 1365 in Lecture Notes on AI, pages 155–176. Springer, 1998.

43. M. d'Inverno and M. Luck. Understanding autonomous interaction. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI 96)*, pages 529–533. John Wiley and Sons, 1996.

44. M. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 31(1):32–64, 1995.

45. M. Endsley. The role of situation awareness in naturalistic decision making. In C. E. Zsambok and G. Klein, editors, *Naturalistic Decision Making.* Lawrence Erlbaum Associates Publishers, 1997.

46. Richard Evans. The future of AI in games. *Game Developer*, August 2001.

47. Xiacong Fan. Towards a building methodology for software agents. In *Object Oriented Information Systems*, pages 45–53, 2000.

48. Jacques Ferber. *Multi-Agent Systems.* Addison Wesley, 1999.

49. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

50. FIPA. *FIPA Communicative Act Library Specification*, 2001.

51. Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Agent Theories, Architectures, and Languages*, pages 21–35, 1996.

52. Richard Gabriel. *Patterns of Software: Tales From the Software Community.* Hungry Minds Inc., 1997.

53. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns.* Addison Wesley, Reading, Massachusetts USA, 1 edition, 1994.

54. X. Gao and L. Sterling. A methodology for building information agents. In Y. Yang, M. Li, and A. Ellis, editors, *Web Technologies and Applications*, pages 43–52. International Academic Publishers, Beijing, China, 1998.

55. Karen Gardner, Alexander Rush, Michael Crist, Robert Konitzer, and Bobbin Teegarden. *Cognitive Patterns.* Managing Object Technology Series. Cambridge University Press, New York, NY, USA, first edition, 1998.

56. J. J. Gibson. The theory of affordances. In R. E. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

57. John J. Gibson. *The Ecological Approach to Visual Perception.* Haughton Mifflin, Boston, first edition, 1979.

58. A. Gomez-Perez, M. Fernandez, and A. De Vicente. Towards a method to conceptualize domain ontologies. In *Workshop on Ontological Engineering at ECAI'96*, pages 41–51, 1996.

59. S. Goss, C. Heinze, M. Papasimeon, A. Pearce, and L. Sterling. The importance of being purposive: Towards reuse in agent oriented information systems. In *Proceedings of the Workshop on Agent Oriented Information Systems (to appear)*, Melbourne, Australia, July 2003.

60. Simon Goss, editor. *Proceedings of the First International Workshop on Team Tactics and Plan Recognition*, 1999.

61. Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

62. T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical report, Knowledge Systems Laboratory, Stanford University, 1992.

63. T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

64. Thomas Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(1):199–220, 1993.

65. Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *KR'91: Principles of Knowledge Representation and Reasoning*, pages 601–602. Morgan Kaufmann, San Mateo, California, 1991.

66. N. Guarino. The ontological level. In B. Smith R. Casati and G. White, editors, *Philosophy and the Cognitive Sciences*. Hlder-Pichler-Tempsky, Kirchberg, Austria, 1994.

67. N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems*, pages 3–15, Trento, Italy, June 1998. IOS Press.

68. R. Harrington. Utilizing bayesian techniques for user interface intelligence. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1996.

69. B. Hayes-Roth, a for, and i systems. Artificial intelligence: Special issue on agents and interactivity, 1995.

70. C. Heinze. Intention oriented analysis: Toward a methodology for agent oriented software engineering. In *In Proceedings of Fifth Australasian Cognitive Science Conference*, Perth, Australia, 2002.

71. C. Heinze, S. Goss, I. Lloyd, and A. Pearce. Collaborating cognitive and sub-cognitive processes for the simulation of human decision making. In *Proceedings of the Third International Simulation Technology and Training Conference, (SimTecT '98)*, Melbourne, Australia, 1998.

72. C. Heinze, S. Goss, and A. Pearce. Plan recognition in military simulation: Incorporating machine learning with intelligent agents. In *Proceedings of IJCAI-99 Workshop on Team Behaviour and Plan Recognition*, pages 53–63, 1999.

73. C. Heinze, M. Papasimeon, and S. Goss. Specifying agent behaviour with use cases. In *Proceedings of Pacific Rim Workshop on Multi-Agents, PRIMA2000*, 2000.

74. C. Heinze, B. Smith, and M. Cross. Thinking quickly: Agents for modeling air warfare. In *Proceedings of the 9th Australian Joint Conference on Artificial Intelligence (AI'98)*, Brisbane, Australia, 1998.

75. C. Heinze and L. Sterling. Using the UML to model knowledge in agent systems. In *AAMAS 2002 Poster Paper 2002*, Bologna, Italy, 2002.

76. Clint Heinze, Martin Cross, Simon Goss, Torgny Josefsson, Ian Lloyd, Graeme Murray, Michael Papasimeon, and Michael Turner. Agents of change: The impact of intelligent agent technology on the analysis of air operations. In L. Jain, N. Ichalkaranje, and G. Tonfoni, editors, *Advances in Intelligent Systems for Defence*, volume 2 of *Series on Innovative Intelligence*, chapter 6, pages 229—264. World Scientific, River Edge, New Jersey, USA, 1 edition, December 2002.

77. Clint Heinze and Leon Sterling. Prepring informtation for agents. In *Proceedings of the Pacific Rim Intelligent Information Agents Workshop (PRIIA 00)*, Melbourne, Australia, 2000.

78. Clinton Heinze, Simon Goss, Torgny Josefsson, Kerry Bennett, Sam Waugh, Ian Lloyd, Graeme Murray, and John Oldfield. Interchanging agents and humans in military simulation. *AI Magazine*, 23(2):37–47, Summer 2002. An earlier version of this paper appeared in the Innovative Applications of AI conference, Seattle, 2001.

79. Rich Hilliard. Using the UML for architectural description. In Robert France and Bernhard Rumpe, editors, *UML'99 - The Unified Modeling Language. Beyond the Standard. Second International Conference, Fort Collins, CO, USA, October 28-30. 1999, Proceedings*, volume 1723, pages 32–48. Springer, 1999.

80. IEEE Computer Society. *3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications: Proceedings.* IEEE, November 1999.

81. Damian Isla and Bruce Blumberg. New challenges for character-based ai for games. In *Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment*, Palo Alto, CA, March 2002.

82. I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison Wesley, Reading, MA, 1992.

83. N. R. Jennings. Agent-based computing. In *Proc. 17th IFIP World Congress on Computing*, Montreal, Canada, 2002.

84. Nicholas R. Jennings. Agent-Oriented Software Engineering. In Francisco J. Garijo and Magnus Boman, editors, *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99)*, volume 1647, pages 1–7. Springer-Verlag: Heidelberg, Germany, 30– 2 1999.

85. Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.

86. L. Jinxin, F. Mark, and S. Bilgic. A requirement ontology for engineering design. *Concurrent Engineering:Research and Applications*, 4(3):279–291, 1996.

87. Chris Johnson. Forensic software engineering and the need for new approaches to accident investigation. Available on-line at http://www.dcs.gla.ac.uk/

88. Michael Jordan and Stuart Russell. Computational intelligence. In *The MIT Encyclopedia of the Cognitive Sciences (MITECS).* MIT Press, 1999.

89. Thomas Juan, Leon Sterling, and Michael Winikoff. Assembling agent oriented software engineering methodologies from features. In *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering*, 2002.

90. G. Kaminka, J. Wendler, and G. Ronen. New challenges in multi-agent intention. In *In Proceedings of the Fall Symposium on Intention Recognition for Collaborative Tasks*, 2001.

91. H. A. Kautz. *A Formal Theory of Plan Recognition and its Implementation*, pages 69–125. Maurgan Kaufmann Publishers, Inc., 1991.

92. H. A. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of 1986 Conference of the American Association for Artificial Intelligence*, 1986.

93. A. Kay. Computer software. *Scientific American*, 251(3):53—59, 1984.

94. Elizabeth A. Kendall. Agent software engineering with role modelling. In *AOSE*, pages 163–170, 2000.

95. Elizabeth A. Kendall, P. V. Murali Krishna, Chirag V. Pathak, and C. B. Suresh. Patterns of intelligent and mobile agents. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 92–99, New York, 9–13, 1998. ACM Press.

96. D. Kinny and M. Georgeff. Modelling and design of multi-agent systems. In J. P. Muller, M. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III*, volume 1193 of *Lecture Notes in AI*. Springer-Verlag, 1997.

97. D. Kinny, M. Georgeff, and A. Rao. A methodology and modelling technique for systems of bdi agents. In *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Mulit-Agent World (MAAMAW'96)*, 1996.

98. Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

99. K. Konolige and M. E. Pollack. A representationalist theory of intention. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 390–395, Chambéry, France, 29– 3 1993. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

100. K. Kostiadis, M. Hunter, and H. Hu. The use of design patterns for the development of multi-agent systems. In *Proceedings IEEE International Conference on Systems, Man, and Cybernetics (SMC2000)*, Tennessee, October 2000.

101. Ryszard Kowalczyk, Seng Wai Loke, Nancy E. Reed, and Graham J. Williams, editors. *Advances in Artificial Intelligence. PRICAI 2000 Workshop Reader, Four Workshops held at PRICAI 2000, Melbourne, Australia, August 28 - September 1, 2000, Revised Papers*, volume 2112 of *Lecture Notes in Computer Science*. Springer, 2001.

102. P. Kruchten. *The Rational Unified Process, An Introduction*. The Addison-Wesley Object Technology Series. Addison-Wesley, 2000.

103. John E. Laird. It knows what you're going to do: adding anticipation to a quakebot. In Jörg P. Müller, Elisabeth Andre, Sandip Sen, and Claude Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 385–392, Montreal, Canada, 2001. ACM Press.

104. Dale A Lambert. Ubiquitous command and control. In Robin Evans, Lang White, Daniel McMichael, and Len Sciacca, editors, *Proceedings of Information Decision and*

Control 99, pages 35–40, Adelaide, Australia, February 1999. Institute of Electrical and Electronic Engineers, Inc.

105. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382—401, July 1982.

106. Jaeho Lee, Marcus J. Huber, Patrick G. Kenny, and Edmund H. Durfee. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS)*, pages 842–849, Houston, Texas, 1994.

107. A. N. Leontiev. The concept of activity in soviet psychology. In J.V. Wertsch, editor, *The problem of activity in psychology*, pages 37–71. ME Sharpe, Inc, 1981.

108. J. Lind. Iterative software engineering for multiagent systems—the massive method. In *Lecture Notes in Artificial Intelligence, LNAI*, New York, 2001. Springer Verlag.

109. Pattie Maes. Artificial life meets entertainment: Lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114, 1995.

110. Kavi Mahesh, Jacquelynn Kud, and Paul Dixon. Oracle at trec8: A lexical approach. In *Proceedings of The Eighth Text REtrieval Conference, (TREC-8)*. National Institute of Standards and Technology, 2000.

111. R. M. Malyankar. A pattern template for intelligent agent systems. In *Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Agents '99*, Seattle, WA, 1999.

112. James Mayfield, Yannis Labrou, and Tim Finin. Desiderata for agent communication languages. In *AAAI Spring Symposium on Information Gathering*, 1995.

113. J. M. McCarthy. Ascribing mental qualities to machines. Technical Report 326, AI Lab, Stanford University, 1979.

114. D. McIlroy and C. Heinze. Advanced operational reasoning for hornet tactics analysis. In *Proceedings of the Second International Simulation Technology and Training Conference, (SimTecT '97)*, Canberra, Australia, 1997.

115. D. McIlroy, C. Heinze, D. Appla, P. Busetta, G. Tidhar, and A. Rao. Towards credible computer generated forces. In *Proceedings of the Second International Simulation Technology and Training Conference, (SimTecT '97)*, Melbourne, Australia, 1997.

116. D. McIlroy, B. Smith, C. Heinze, and M. Turner. Air defence operational analysis using the swarmm model. In *Proceedings of the Asia Pacific Operations Research Symposium (APORS'97)*, Melbourne, Australia, 1997.

117. Jonathan C. McPhail and Dwight Deugo. Deciding on a pattern. *Lecture Notes in Computer Science*, 2070, 2001.

118. T. Menzies, S. Waugh, S. Goss, and Robert F. Cohen. Evaluating a temporal causal ontology. In *Proceedings of the Conference on Formal Ontologies in Information Systems (FOIS '97)*, 1997.

119. M. Minsky. *The Society of Mind.* Simon and Schuster, New York, 1986.

120. M. Minsky. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. *AI Magazine*, 1991.

121. R. J. Mooney. Learning plan schemata from observation: Explanation-based learning for plan recognition. *Cognitive Science*, 14:483–509, 1990.

122. S. Mulgund, G. Rinkus, C. Illgen, and J. Friskie. Olipsa: On-line intelligent processor for situation assessment. In *In Proceedings of the Second Annual Symposium and Exhibition on Situational Awareness in the Tactical Air Environment*, 1997.

123. D. Ndumu and H. Nwana. Research and development challenges for agent-based systems. *IEEE Proceedings On Software Engineering*, 144(1):p2–10, 1997.

124. A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, Massachusetts, 1990.

125. Alan Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.

126. E. Norling and C. Heinze. Naturalistic decision making and agent oriented cognitive modelling. In *Proceedings of Fifth Australasian Cognitive Science Conference*, 2000.

127. D. A. Norman. *The psychology of everyday things.* Basic Books, New York, 1988.

128. D. A. Norman. *The design of everyday things.* Doubleday, New York, 1990.

129. H. S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244, 1995.

130. Object Modeling Group. *Unified Modelling Language Summary*, version 1.1 edition, September 1997.

131. J. ODell. Agent technology green papaer. Agent Working Group OMG Document 1.0, Object Modeling Group, August 2000.

132. James Odell, H. Van Dyke Parunak, and Bernhard Bauer. Extending UML for Agents. In *Proceedings of AOIS Workshop at AAAI*, 2000.

133. James Odell, H. Van Dyke Parunak, Mitch Fleischer, and Sven Breuckner. Modeling agents and their environment. In *Proceedings of the 4th International Workshop on Agent Orinted Software Engineering (AOSE '02)*, Bologna, Italy, 2002.

134. Patterns Home Page. http://hillside.net/patterns/. Web, December 2000.

135. M. Papasimeon. Designing environments for agents. In *Proceedings of Fifth Australasian Cognitive Science Conference*, Perth, Australia, 2002.

136. Michael Papasimeon and Clint Heinze. Extending the UML for designing jack agents. In *Proceedings of the Australian Software Engineering Conference, (ASWEC 01)*, Canberra, Australia, 2001.

137. A. R. Pearce. *Relational Evidence Theory and Spatial Interpretation Procedures.* Ph.d. thesis, Curtin University, School of Computing, Perth Australia, 1997.

138. A. R. Pearce and T. Caelli. Interactively matching hand-drawings using induction. *Computer Vision and Image Understanding*, 73(2), 1999.

139. A. R. Pearce, T. Caelli, and S. Goss. On learning spatio-temporal structures in two different domains. *Lecture Notes in Artificial Intelligence*, 1352(2):551–558, 1998.

140. R. Pew and A. Mavor. *Modeling Human and Organizational Behavior: Applications to Military Simulations.* National Academy Press, Washington, 1998. Final report of the National Research Council Panel on Modeling Human Behavior and Command Decision Making: Representations for Military Simulations.

141. Isabella Poggi, Catherine Pelachaud, and Fiorella de Rosis. Eye communication in a conversational 3d synthetic agent. *AI Communications*, 13(3):169–182, 2000.

142. S. Poslad, P. Buckle, and R. Hadingham. The fipa-os agent platform: Open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, pages 355–368, 2000.

143. Wolfgang Pree. Meta patterns — A means for capturing the essentials of reusable object-oriented design. *Lecture Notes in Computer Science*, 821:150–162, 1994.

144. D. Pynadath. *Probabilistic Grammars for Plan Recognition.* Phd thesis, University of Michigan, 1999.

145. D. Pynadath, M. Tambe, Y. Arens, H. Chalupsky, Y. Gil, C. Knoblock, H. Lee, K. Lerman, J. Oh, S. Ramachandran, P. Rosenbloom, and T. Russ. Electric elves: Immersing an agent organization in a human organization. In *In Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents — the human in the loop, 2000*, 2000.

146. A. Rao. A unified view of plans as recipes. Technical Report 77, Australian Artificial Intelligence Institute, Melbourne, Australia, August 1997.

147. A. S. Rao. Means-end plan recognition—towards a theory of reactive recognition. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 497–508, Bonn, FRG, May 1994. Morgan Kaufmann Publishers; San Francisco, CA, USA.

148. A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. In *Second International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, CA, 1991.

149. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

150. A. S. Rao and M. P. Georgeff. Formal models and decision procedures for multi-agent systems. Technical Report Technical Note 61, Australian AI Institute, 171 La Trobe Street, Melbourne, Australia, 1995.

151. Anand S. Rao and Graeme Murray. Multi-agent mental-state recognition and its application to air-combat modeling. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence (DAI-94)*, pages 283–304, Seattle, WA, USA, 1994.

152. J. Rasmussen, A. Pejtersen, and L. Goodstein. *Cognitive Systems Engineering.* Wiley Interscience, 1994.

153. Philip R.Cohen, , and H.J.Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42((2–3)):213–361, 1990.

154. Dirk Riehle and Heinz Zullighoven. *Understanding and Using Patterns in Software Development.* Hungry Minds Inc., 1997.

155. D. Rosenberg and K. Scott. *Use Case Driven Object Modeling with UML.* Addison Wesley, 1999.

156. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

157. C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, 1992. Morgan Kaufmann.

158. Robert Schwartz. *Vision.* Blackwell Publishers, Cambridge, Ma, USA, 1994.

159. J. R. Searle. The intentionality of intention and action. *Cognitive Science*, 1(4):47–70, 1980.

160. Wilfrid Sellars. Empiricism and the philosophy of mind. In Herbert Feigl and Michael Scriven, editors, *The Foundations of Science and the Concepts of Psychology and Psychoanalysis*, volume 1, pages 253–329. University of Minnesota Press, Minneapolis, 1956.

161. Herbert Simon. *The Sciences of the Artificial.* MIT Press, 1996.

162. Liz Sonenberg and Gil Tidhar. Observations on team-oriented mental state recognition. In *Proceedings of the 1999 IJCAI Workshop on Team Modelling and Plan Recognition*, 1999.

163. William Song. Web document modelling and clustering. In *Proceedings of the ER'97 Workshop on Conceptual Modeling in Multimedia Information Seeking*, 1997.

164. Salvatore J. Stolfo, Andreas L. Prodromidis, Shelley Tselepis, Wenke Lee, Dave W. Fan, and Philip K. Chan. JAM: Java agents for meta-learning over distributed databases. In *Knowledge Discovery and Data Mining*, pages 74–81, 1997.

165. L. Suchman. *Plans and situated actions: the problem of human/machine communication.* Cambridge University Press, 1987.

166. M. Tambe. Agent architectures for flexible, practical teamwork. In *National Conference on Artificial Intelligence (AAAI-97)*, 1997.

167. T. Thom. *The Flying Training Manual - Pre-flight Briefings and Air Exercises.* Aviation Theory Centre Pty. Ltd., Williamstown, Australia, 1993.

168. G. Tidhar, P. Busetta, and C. Heinze. Intention recognition in air combat. Technical report, Defence Science and Technology Organisation, 1999.

169. G. Tidhar, C. Heinze, S. Goss, G. Murray, D. Appla, and I. Lloyd. Using intelligent agents in military simulation or "using agents intelligently". In *Proceedings of the Eleventh Innovative Applications of Artificial Intelligence Conference*, 1999.

170. G. Tidhar, C. Heinze, and M. Selvestrel. Flying together: Modelling air misson teams. *Journal of Applied Intelligence*, 8(3):195–218, 1998.

171. G. Tidhar, M. Selvestrel, and C. Heinze. Modelling teams and team tactics in whole air mission modelling. In G. F. Forsyth and M. Ali, editors, *Proceedings of the Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-95)*, pages 373–381, Melbourne, Australia, June 1995.

172. Gil Tidhar. *Organisation Oiented Programming.* Phd thesis, University of Melbourne, Melbourne, Australia, 2000.

173. M. T. Turvey and T. E. Shaw. Toward an ecological physics and a physical psychology. In R. L. Solso and D. W. Massaro, editors, *The Science of the Mind*, pages 144–172. Oxford University Press, New York, 1995.

174. Sun Tzu. *The Art of War.* Shambala Publications, 1988.

175. Mike Uschold. Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.

176. Hans Van Vliet. *Software Engineering: Principles and Practice.* John Wiley and Sons, 2002.

177. H. von Helmholtz. *Helmholtz's Physiological Optics.* Haughton Mifflin, Boston, 3 edition, 1924.

178. B. Webster. *Pitfalls of Object Oriented Development.* Hungry Minds Inc., 1995.

179. D. Weerasooriya, A. Rao, and K. Ramamohanarao. Design of a concurrent agent-oriented language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890)*, pages 386–402. Springer-Verlag: Heidelberg, Germany, 1995.

180. Gio Wiederhold. Mediators in the architecture of future information systems. *Computer Magazine of the Computer Group News of the IEEE Computer Group Society*, 1992.

181. M. Wilson, P. Barnard, T. Green, and A. MacLean. Knowledge based task analysis for human computer systems. In G. C. van der Veer, T.R.G. Green, J. Hoc, and D. M. Murray, editors, *Working with Computers Theory verses Outcome*, pages 47–87. Academic Press, London, 1988.

182. Michael Winikoff, Lin Padgham, and James Harland. Simplifying the development of intelligent agents. In *Australian Joint Conference on Artificial Intelligence*, pages 557–568, 2001.

183. Wittgenstein. *Tractatus Logico Philosophicus.* Routledge, 2001.

184. M. Wooldridge. Agent-based software engineering. In *IEEE Proceedings on Software Engineering*, pages 22–37, 1997.

185. M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

186. Michael Wooldridge. Intelligent agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–78. The MIT Press, Cambridge, MA, USA, 1999.

187. Michael Wooldridge and Paolo Ciancarini. Agent-oriented software engineering: The state of the art. In *AOSE*, pages 1–28, 2000.

188. Michael Wooldridge and Nicholas R. Jennings. Pitfalls of agent-oriented development. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 385–391, New York, 9–13, 1998. ACM Press.

189. M.J. Wooldridge. Agent-based software engineering. *IEEE Proceedings on Software Engineering*, 144(1):26–37, February 1997.

190. Qiang Yang, Irene Abi-Zeid, and Luc Lamontagne. An agent for intelligent situation assessment. In *Artificial Intelligence: Methodology, Systems, Applications*, pages 466–474, 1998.

191. Philip D. Zelazo, David R. Olson, and Janet W. Astington. *Developing Theories of Intention: Social Understanding and Self Control.* Lawrence Erlbaum, 1999.

192. W. Zhang and R. W. Hill. A template-based and pattern-driven approach to situation awareness and assessment in virtual humans. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 116–123, Barcelona, Catalonia, Spain, 2000. ACM Press.

193. F. Zini and L. Sterling. Designing ontologies for agents. In *Proceedings of Appia–Gulp–Prode '99*, L'Aquila, Italy, September 1999.

194. Zsambok and Klein. *Naturalistic Decision Making.* Lawrence Erlbaum Associates Publishers, 1997.

DISTRIBUTION LIST

Modelling Intention Recognition for Intelligent Agent Systems

Clint Heinze

	Number of Copies
DEFENCE ORGANISATION	
Task Sponsor	
DSSL	1
S&T Program	
Chief Defence Scientist	1
FAS Science Policy	
AS Science Corporate Management	
Director General Science Policy Development	
Counsellor, Defence Science, London	Doc Data Sheet
Counsellor, Defence Science, Washington	Doc Data Sheet
Scientific Adviser to MRDC, Thailand	Doc Data Sheet
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc Data Sheet and Dist List
Scientific Adviser, Army	Doc Data Sheet and Dist List
Air Force Scientific Adviser	Doc Data Sheet and Exec Summ
Scientific Adviser to the DMO M&A	Doc Data Sheet and Dist List
Scientific Adviser to the DMO ELL	Doc Data Sheet and Dist List
Systems Sciences Laboratory	
CAOD	Doc Data Sheet and Dist List
RLAOR, AOD	1
Dr Simon Goss, AOD	2
Mr Graeme Murray, AOD	1
Mr Michael Papasimeon, Air Operations Division	1
Ms Jennifer Sandercock, Air Operations Division	1
Dr Clint Heinze, Air Operations Division	4
DSTO Library and Archives	
Library, Fishermans Bend	Doc Data Sheet
Library, Edinburgh	1 and Doc Data Sheet

Library, Sydney	Doc Data Sheet
Library, Stirling	Doc Data Sheet
Library, Canberra	Doc Data Sheet
Defence Archives	1
Capability Development Group	
Director General Maritime Development	Doc Data Sheet
Director General Capability and Plans	Doc Data Sheet
Assistant Secretary Investment Analysis	Doc Data Sheet
Director Capability Plans and Programming	Doc Data Sheet
Director Trials	Doc Data Sheet
Chief Information Officer Group	
Deputy Chief Information Officer	Doc Data Sheet
Director General Information Policy and Plans	Doc Data Sheet
AS Information Strategy and Futures	Doc Data Sheet
AS Information Architecture and Management	Doc Data Sheet
Director General Australian Defence Simulation Office	Doc Data Sheet
Director General Information Services	Doc Data Sheet
Strategy Group	
Director General Military Strategy	Doc Data Sheet
Director General Preparedness	Doc Data Sheet
Assistant Secretary Strategic Policy	Doc Data Sheet
Assistant Secretary Governance and Counter-Proliferation	Doc Data Sheet
Navy	
SO (SCIENCE), COMAUSNAVSURFGRP, NSW	Doc Data Sheet
Director General Navy Capability, Performance and Plans, Navy Headquarters	Doc Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters	Doc Data Sheet
Deputy Director (Operations) Maritime Operational Analysis Centre, Building 89/90, Garden Island, Sydney Deputy Director (Analysis) Maritime Operational Analysis Centre, Building 89/90, Garden Island, Sydney	Doc Data Sheet and Dist List
Army	
ABCA National Standardisation Officer, Land Warfare Development Sector, Puckapunyal	Doc Data Sheet (pdf format)
SO (Science), Deployable Joint Force Headquarters (DJFHQ)(L), Enoggera QLD	Doc Data Sheet

SO (Science), Land Headquarters (LHQ), Victoria Barracks, NSW	Doc Data Sheet and Exec Summ
Air Force	
SO (Science), Headquarters Air Combat Group, RAAF Base, Williamtown	Doc Data Sheet and Exec Summ
Joint Operations Command	
Director General Joint Operations	Doc Data Sheet
Chief of Staff Headquarters Joint Operation Command	Doc Data Sheet
Commandant ADF Warfare Centre	Doc Data Sheet
Director General Strategic Logistics	Doc Data Sheet
Intelligence and Security Group	
DGSTA, Defence Intelligence Organisation	1
Manager, Information Centre, Defence Intelligence Organisation	1 (pdf format)
Assistant Secretary Capability Provisioning	Doc Data Sheet
Assistant Secretary Capability and Systems	Doc Data Sheet
Assistant Secretary Corporate, Defence Imagery and Geospatial Organisation	Doc Data Sheet
Defence Materiel Organisation	
Deputy CEO, DMO	Doc Data Sheet
Head Aerospace Systems Division	Doc Data Sheet
Head Maritime Systems Division	Doc Data Sheet
Chief Joint Logistics Command	Doc Data Sheet
Head Materiel Finance	Doc Data Sheet
Defence Libraries	
Library Manager, DLS-Canberra	Doc Data Sheet
Library Manager, DLS-Sydney West	Doc Data Sheet
UNIVERSITIES AND COLLEGES	
Australian Defence Force Academy Library	1
Head of Aerospace and Mechanical Engineering, ADFA	1
Deakin University Library, Serials Section (M List), Geelong, Vic	1
Hargrave Library, Monash University	Doc Data Sheet
Librarian, Flinders University	1
OTHER ORGANISATIONS	
National Library of Australia	1

NASA (Canberra)	1
Library of New South Wales	1

INTERNATIONAL DEFENCE INFORMATION CENTRES

US - Defense Technical Information Center	2
UK - Dstl Knowledge Services	2
Canada - Defence Research Directorate R&D Knowledge and Information Management (DRDKIM)	1 (pdf format)
NZ - Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

INFORMATION EXCHANGE AGREEMENT PARTNERS

National Aerospace Laboratory, Japan	1
National Aerospace Laboratory, Netherlands	1

SPARES

DSTO Edinburgh Library	5

Total number of copies:	42

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA				1. CAVEAT/PRIVACY MARKING
2. TITLE Modelling Intention Recognition for Intelligent Agent Systems			3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)	
4. AUTHOR Clint Heinze			5. CORPORATE AUTHOR Systems Sciences Laboratory PO Box 1500 Edinburgh, South Australia, Australia 5111	
6a. DSTO NUMBER DSTO–RR–0286	6b. AR NUMBER 013-251		6c. TYPE OF REPORT Research Report	7. DOCUMENT DATE November, 2004
8. FILE NUMBER 2004/1012868/1	9. TASK NUMBER LRR 03/226	10. SPONSOR DSSL	11. No OF PAGES 217	12. No OF REFS 194
13. URL OF ELECTRONIC VERSION http://www.dsto.defence.gov.au/corporate/reports/DSTO–RR–0286.pdf			14. RELEASE AUTHORITY Chief, Air Operations Division	

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved For Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DEFTEST DESCRIPTORS

Psychology	Perception
Recognition	Visual perception
Software agents	Intelligent systems
Artificial intelligence	Flight simulation

19. ABSTRACT

Mainstream visual psychology presents a 'sense then infer' account of vision that is analogous to the 'sense then infer' processing that characterises the agent intention recognition literature. From ecological psychology comes Gibson's theory of visual perception that highlights the importance of the environment in explaining the nature of vision and recognition and claims that higher order structures are directly accessible. This theory can be used as the stepping-off point for an account of intention recognition and the means by which it might be modelled. Furthermore, the capacity for virtual environments to be designed 'agent friendly' provides yet another dimension of design freedom. When accompanied by an explicit model of perception the intention recognition problem can be cast as a software design problem. The resulting design patterns provide useful options for modelling intention recognition in intelligent agent systems.